Journal of Information and Communication Technology

Volume 16, Issue 61-62, Autumn and Winter 2024, pp. 1-19

An Overview on Replica Consistency Methods in Distributed Systems and Future Works

Mahsa Beigrezaei[*1]

[1] Department of Computer Engineering, Yadegar-e-Imam Khomeini (RAH) Shahre Rey Branch, Islamic Azad University, Tehran, Iran.

Received: 07 May 2023, Revised: 27 November 2023, Accepted: 01 December 2023

Paper type: Research

Abstract

Nowadays, applications generate huge amounts of data, in the range of several terabytes or petabytes. This data is shared among many users around the world. Distributed systems such as grid and cloud provide a suitable platform for these applications, enabling the use of these diverse mass data applications in a distributed manner. In these systems, they use data replication to face performance problems, guarantee service quality, and increase data accessibility. Replication, despite its many advantages, also brings administrative costs. The balance between the consistency cost of replication and the benefits of replication is a hotly debated topic among researchers in this field. Therefore, paying attention to the consistency of replication plays an effective role in the efficiency of these systems. Many strategies have been proposed by researchers in the field of data replication consistency. Each of these strategies try to reduce consistency costs and provide effective solutions in this field by considering various parameters such as read rate, write rate, old data tolerance rate, number of replicas and communication bandwidth in determining the consistency levels of replicas. In this article, we will examine the concepts related to replication and replica consistency and categorize its types and review previous works in this field. The done works have been compared from the perspective of system type, decision parameters, compatibility model and improved parameters. At the end, the open issues in this field are raised.

Keywords: Data Replication, Consistency, Models, Cloud, IOT, Distributed System.

[*] Corresponding Author's email: m.beigrezai@gmail.com

دوفصلنامه
فناوری اطلاعات و ارتباطات ایران

سال شانزدهم، شماره‌های ۶۱ و ۶۲، پاییز و زمستان ۱۴۰۳، صفحه ۱ الی ۱۹

مروری بر سازگاری تکثیر داده در سیستم‌های توزیع شده

مهسا بیگ رضایی[1*]

[1] گروه مهندسی کامپیوتر، واحد یادگار امام خمینی(ره) شهرری، دانشگاه آزاد اسلامی، تهران، ایران

تاریخ دریافت: ۱۴۰۲/۰۲/۱۷ تاریخ بازبینی: ۱۴۰۲/۰۹/۰۶ تاریخ پذیرش: ۱۴۰۲/۰۹/۱۰
نوع مقاله: پژوهشی

چکیده

سیستم‌های توزیع شده مانند گرید و ابر به منظور مواجهه با مشکلات کارایی، تضمین کیفیت خدمت و افزایش دسترسی‌پذیری به داده‌ها از تکثیر داده استفاده می‌کنند. تکثیر با وجود مزایای بسیار هزینه‌های مدیریتی نیز به همراه دارد. سازگار نگه داشتن تکثیرها از جمله مهم‌ترین هزینه‌های ناشی از تکثیر است. تعادل بین هزینه سازگاری تکثیر و مزایای تکثیر یک موضوع مورد بحث و داغ در بین محققان این حیطه است. لذا توجه به سازگاری تکثیر نقش موثری در کارایی این سیستم‌ها بازی می‌کند. استراتژی‌های بسیاری توسط محققان در حیطه سازگاری تکثیر داده ارائه شده است. هر کدام از این استراتژی‌ها با در نظر گرفتن پارامترهای مختلفی مانند نرخ خواندن، نرخ نوشتن، نرخ تحمل داده‌های قدیمی، تعداد تکثیرها و پهنای باند ارتباطی در تعیین سطوح سازگاری تکثیرها سعی در کاهش هزینه‌های سازگاری و ارائه راهکارهای مؤثر در این حوزه دارند. در این مقاله به مفاهیم تکثیر و سازگاری تکثیر پرداخته می‌شود. دسته‌بندی‌ها و روش‌های سازگاری موجود در این حوزه بررسی می‌شود. کارهای انجام شده در حیطه سازگاری تکثیر داده از دیدگاه‌های مختلفی مانند نوع سیستم، پارامترهای تصمیم‌گیری، ابزار شبیه‌سازی، مدل سازگاری و پارامترهای بهبود داده شده مقایسه می‌شوند. همچنین در پایان، موضوعات باز در این حوزه مطرح می‌شود.

کلیدواژگان: تکثیر داده، سازگاری، مدل‌های سازگاری، سیستم توزیعی، ابر، گرید داده.

* رایانامهٔ نویسنده مسؤول: m.beigrezai@gmail.com

۱- مقدمه

امروزه همراه با پیشرفت و رشد اینترنت، برنامه‌های کاربردی با تعداد زیادی کاربر توزیع شده در سطح جهان تمایل به ارتباط برخط دارند. در این راستا سیستم‌های توزیع شده مانند خوشه‌ها[1]، گرید محاسباتی، ابر محاسباتی و سیستم‌های نظیر به نظیر[2] پا به عرصه وجود گذاشتند. افزایش کاربران و ازدحام درخواست‌ها و دوری کاربران از منابع، کاهش کیفیت خدمت و تأخیر و نارضایتی کاربران را به همراه دارد. این کاستی‌ها زمانی که درخواست کاربران نیازمند حجم زیادی از داده‌ها باشند، به طور چشمگیری افزایش می‌یابد.

برنامه‌های کاربردی امروزی در بسیاری از زمینه‌ها مثل تجاری، علمی، آموزشی سرگرمی در حال تولید حجم انبوهی اطلاعات است [۱-۵]. به عنوان مثال، برنامه‌ها با کاربرد فیزیک با انرژی بالا[3] [۶] و بیوانفورماتیک و وب سایت‌های محبوب مانند Facebook, Twitter, Google, Amazon در شبکه‌های گرید، شبکه‌های ابری [۷]، اینترنت اشیا (IOT) [۸] تولیدکننده داده با حجم بالا، در حد ترابایت و پتابایت است. با رشد روز افزون این برنامه‌ها در سیستم‌های توزیع شده، تضمین فاکتورهای کارایی مانند مقیاس پذیری، قابلیت اطمینان، دسترسی‌پذیری، قابلیت گسترش، کاهش تأخیر و هزینه‌ها، یکی از مشکلات اصلی این محیط‌های توزیع شده است. تکثیر داده یک روش کلیدی و کارا برای مقابله با مشکل‌های مطرح شده و رسیدن به تحمل‌پذیری خطا و حل مشکل گلوگاه، است. تکثیر داده تکنیکی است که چندین کپی از داده مورد نیاز کاربر را در سطح سیستم توزیعی در نزدیکی کاربر به منظور افزایش کارایی قرار می‌دهد. امروزه استراتژی تکثیر به صورت گسترده و برای افزایش کارایی در شبکه جهانی وب، شبکه‌های نظیر به نظیر [۹]، سنسورها، ابر محاسباتی، گرید [۱۰] و سایر سیستم‌های توزیعی استفاده می‌شود [۱۱- ۱۵].

امروز با ظهور سیستم‌های توزیع شده‌ی نوین مانند ابر، تکثیر داده دوباره به عنوان موضوعی داغ ِمورد توجه محققان قرار گرفته است. در این سیستم‌ها، برای رسیدن به کارایی مد نظر داده‌های تعداد زیادی از برنامه‌های کاربردی تکثیرمی‌شود. این سیستم‌ها استراتژهای مختلف تکثیر و مدیریت سازگاری تکثیر را به کار می‌برند. در استراتژی‌های تکثیر داده، چندین کپی از داده در سرورهای مختلف در مناطق مختلف توزیع می‌شود. این کار به منظور کاهش زمان دسترسی و مصرف پهنای باند، توزیع بار درخواست‌ها و زنده نگه داشتن خدمات داده در زمان خرابی سرورها است [۱۶, ۱۷] و با وجود این مزایا، تکثیر سربارهایی را به سیستم توزیعی تحمیل می‌کند که در صورت عدم کنترل مناسب این سربار هزینه‌های ناشی شده از آن بیشتر از سودمندی آن خواهد بود. سازگار نگه داشتن تکثیرها یکی از مهم‌ترین و پر چالش‌ترین هزینه-های مدیریت تکثیر داده است. مدیریت سازگاری تکثیرها اگر به درستی انجام نشود می‌تواند منجر به کندی سیستم شود. به طوری که تکثیر را تبدیل به معظلی بزرگ در سیستم‌های توزیعی کند. از این رو مدیریت سازگاری تکثیر در سیستم‌های توزیعی با قابلیت خواندن و نوشتن بالا بسیار اهمیت دارد.

با وجود اهمیت مبحث مدیریت سازگاری تکثیر داده، تعداد کمی از مقالات مروری پیشین به طور ویژه به بررسی این مبحث پرداخته‌اند. بر اساس مطالعات ما، مرجع‌های [۱۸-۲۱] مقاله‌های مروری موجودی هستند که به مبحث سازگاری تکثیر داده پرداخته‌اند. در این مرجع‌ها به طور جامع به مروری بر این مبحث پرداخته نشده است. به عنوان نمونه در [۱۸]، تنها به مدیریت سازگاری در شرایط خاص تکثیر خوشبینانه داده پرداخته شده است. در مقاله [۱۹]، تنها در سیستم توزیعی ابر مدیریت سازگاری مورد بررسی قرار گرفته است و تنها به تعدادی محدودی از روش‌های پایه و قدیمی پرداخته شده است. در [۲۱]، تنها به تعدادی کارهای ارائه شده در مبحث الگوریتم‌های سازگاری در سیستم توزیعی گرید پرداخته شده است. تقسیم‌بندی‌های ارائه شده در این مقاله از نقطه نظر جنبه توپولوی‌های گرید و پارامترهای ارزیابی و نرم‌افزار شبیه‌سازی است. در [۲۰]، پرتوکل‌های پایه تکثیر و پروتکل‌های به‌روزرسانی تکثیر در سیستم‌های پایگاه داده‌ای بررسی می‌شود. در این مقاله کارهای جدید کمی مطرح شده و بیشتر تمرکز بر روی کارهای پایه است.

از بررسی کارهای مروری انجام شده می‌توان متوجه شد در هیچکدام از کارهای پیشین تقسیم‌بندی‌ها در تمام سیستم‌های توزیعی بررسی نشده، و کارهای آینده و چالش‌ها و موضوعات باز به طور خاص بررسی نشده‌اند. در این مقاله برای رفع کاستی‌های موجود الگوریتم‌های سازگاری تکثیر را در سیستم‌های توزیعی قدیمی و جدید بررسی می‌کنیم. در این برررسی در ابتدا مفهوم تکثیر و تقسیم‌بندی‌های تکثیر و مفهوم سازگاری تکثیر می‌پردازیم. در ادامه به برررسی روش‌های پایه و مشهور و تقسیم‌بندی‌های موجود و جدید از منظرهای مختلف پرداخته می‌شود. در بخش‌هایی نیز به طور جدا کارهای موجود و جدید در سیستم توزیعی‌های محبوب بررسی می‌شود. در این مقاله سعی می‌شود تقسیم‌بندی‌ها به طور

[1] Clusters
[2] Peet to Peer
[3] High Energy Physics (HEP)

جامع‌تری مورد توجه قرار گیرد. در پایان نیز چالش‌ها و کارهای آتی بیان می‌شود تا دید جامع از آینده این حوزه به خواننده داده شود. این مقاله شامل ۷۵ مرجع است که در یک دوره زمانی منتشر شده است. این مقالات شامل کارهای انجام شده توسط محققان از دهه ۱۹۸۰ تا نیمه سال ۲۰۲۳ می‌شود. با توجه به ماهیت آثار تحلیل شده، این مقاله هم مقالات کنفرانس‌ها و هم مجلات، را پوشش می‌دهد به طور خاص، این تحلیل ۱۳ کنفرانس، ۵۰ مجله، هشت یادداشت سخنرانی، یک مرجع آنلاین، و سه کتاب را در نظر پوشش می‌دهد. در بین مجلات، بیشتر آنها با موارد نمایه شده در گزارش استنادی تامسون مجله مطابقت دارد که به عنوان شاخصی مناسب برای اندازه گیری تأثیر مجلات در نظر گرفته می‌شود.

در ادامهٔ این مقاله در بخش ۲ به بررسی تکثیر داده و تقسیم‌بندی‌ها در تکثیر داده می‌پردازیم. در بخش ۳ مفهوم سازگاری، روش‌های مطرح در سازگای، تقسیم‌بندی‌ها و مدل‌های سازگای موجود مطرح می‌شود. در بخش ۴ سازگاری در سیستم‌های توزیعی ابر و مه[1] و اینترنت اشیا مورد بررسی قرار خواهد گرفت. در بخش ۵ چالش‌های موجود و کارهای آتی مطرح شده و در بخش پایانی نتیجه‌گیری بیان می‌شود.

۲- تکثیر داده و تقسیم‌بندی الگوریتم‌های تکثیر

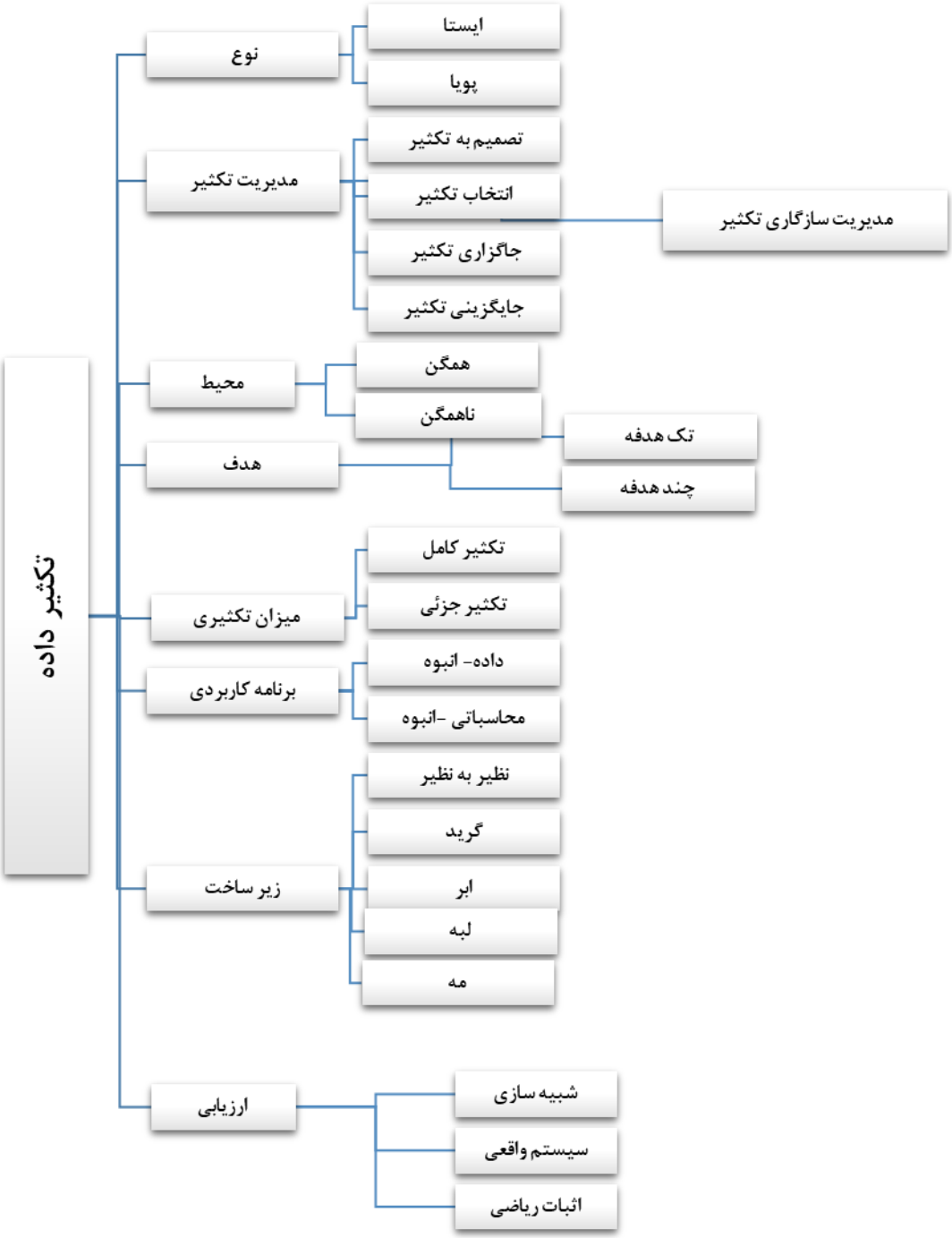
با وجود مزایای بسیار ذکر شده، تکثیر سربارهایی را به سیستم توزیعی تحمیل می‌کند. در صورت عدم کنترل مناسب این سربار هزینه‌ها بیشتر از سودمندی آن خواهد بود. چرا که عمل تکثیر ذاتاً عملی هزینه بردار است و در جهت استفاده از آن منابع ذخیره‌سازی و پهنای‌باند شبکه باید مصرف شود. لذا مدیریت تکثیر، امری خطیر و چالشی مهم در سیستم‌های توزیع شده است. در مدیریت تکثیرهای داده باید به سوال‌های مهمی در این بخش پاسخ داده شود. این سؤال‌ها عبارت‌اند از:۱. چه فایلی باید تکثیر شود؟ ۲. چه زمانی عمل تکثیر انجام شود؟ ۳. بهترین مکان تکثیر کجاست؟ ۴. چه تکثیرهای باید حذف شوند؟ ۵. چگونه داده‌ها سازگار نگه داشته شوند؟ چهار سؤال اول توسط الگوریتم‌های تکثیر و سؤال آخر توسط مدل‌ها و استراتژی‌های سازگاری پاسخ داده می‌شود [۲۲]. در این مقاله تمرکز اصلی بر روی سوال پنجم است. در واقع این مقاله به بررسی روش‌ها و کارهای انجام شده در پاسخ به سوال پنجم می‌پردازد. سایر سوال‌ها تمرکز این مقاله مروری نمی‌باشد.

در شکل ۱ تکثیر داده‌ها را به طور خلاصه از دیدگاه‌های مختلف بررسی و طبقه‌بندی می‌کنیم. همانطور که در شکل ۱ نشان داده شده است و روش‌های تکثیر به طور کلی به الگوریتم‌های پویا و ایستا طبقه‌بندی می‌شوند [۲۳-۲۲]. در الگوریتم‌های ایستا، تعداد تکثیرها و مکان آنها ثابت است. در مقابل الگوریتم‌های تکثیر پویا با الگوهای دسترسی کاربر و تغییرات سازگار هستند. الگوریتم‌های پویا تغییر نیازمندی‌های سیستم را در نظر می‌گیرند. ماهیت محیط توزیع شده پویا است. بنابراین روش‌های پویا موفق‌تر از ایستا در سیستم‌های توزیع شده واقعی هستند. در شبکه‌های پراکنده بزرگ به دلیل فاصله جغرافیایی، تأخیر دسترسی به داده‌ها زیاد است. تأخیرهای زیاد ممکن است منجر به یک مشکل حیاتی برای ارائه‌دهندگان خدمات و مشتریان شود [۱۷]. از نظر نوع سازمان تصمیم‌گیری می‌توان آن را به دو دسته توزیعی و متمرکز طبقه‌بندی کرد. در این دو روش، چندین گره توزیع شده و یک گره به ترتیب تصمیمات تکثیری را اتخاذ می‌کنند. از سوی دیگر، فرآیند تصمیم‌گیری می‌تواند به صورت برخط یا برون خط انجام شود. الگوریتم‌ها می‌توانند یک هدف یا چند هدف داشته باشند. علاوه بر این، بر اساس درصد تکثیر فایل، تکثیر داده‌ها را می‌توان به تکثیر کامل و تکثیر جزئی تقسیم کرد. در تکثیر کامل ۱۰۰٪ حجم فایل کپی شده و به صورت جزئی، فقط یک قسمت ضروری از فایل کپی می‌شود. در تکثیر جزئی می‌توان از دانه‌بندی‌های مختلف تکثیر مانند بلوک، جدول، ردیف، شی و پایگاه داده استفاده کرد. اغلب تکثیر جزئی مؤثرتر از تکثیر کامل است; اما پیچیده‌تر است. اصلی ترین زیر ساخت‌ها در سطح بزرگ مقیاس که از تکثیر داده استفاده می‌کنند شبکه‌های نظیر به نظیر، گرید، ابر، لبه و مه هستند. محیط هایی که از تکثیر داده بهره می‌برند می‌توانند همگن یا ناهمگن باشند. تاکنون الگوریتم‌های تکثیر زیادی ارائه شده است. ارزیابی کارایی این الگوریتم‌ها توسط روش‌های مختلفی مانند شبیه‌سازی، اثبات ریاضی، پیاده‌سازی در محیط واقعی انجام شده است [۲۴].

۳- سازگاری تکثیرهای داده

بحث سازگاری تکثیر داده در جایگاه‌های مختلفی مثل معماری کامپیوتر، پایگاه داده و سیستم‌های توزیع شده مطرح می‌شود که در هر کدام می‌تواند معانی متفاوتی داشته باشد. در تمام آنها باید به این سؤال پاسخ داده شود که چه رخدادی باید بیفتد، زمانی که تکثیری که احتمالاً به زودی مورد دسترس واقع می‌شود، تغییر کند. بسته به محتوای داده تکثیر شده و نوع برنامه کاربردی که تکثیر به آن وابسته است پاسخ‌های متفاوتی به این سؤال داده می‌شود که تحت عنوان سازگاری به این مقوله پرداخته می‌شود.

[1] Fog



شکل ۱. تقسیم‌بندی الگوریتم‌های تکثیر داده از دیدگاهای مختلف

این پاسخ در سیستم‌های توزیع شده اهمیت بالایی دارد [۲۴]. چرا که باید توسعه‌دهنده‌ها از این سؤال پایه‌ای آگاه باشند. به خصوص زمانی که مشتری‌ها انسان نباشند، که این مسئله در سیستم‌های توزیعی امروزی مانند ابر، گرید و نظیر به نظیر یک اتفاق رایج است. در تعریفی دیگر، سازگاری تکثیر داده یعنی معتبر بودن، دقت، قابل استفاده بودن و صحت داده‌های تکثیر شده از نظر برنامه‌هایی که از این داده‌ها استفاده می‌کنند [۱۲, ۲۱]. روش‌های مختلفی برای رسیدن به این تعاریف مطرح شده است. یکی از بخش‌های مهم در مدیریت تکثیر داده، سازگاری تکثیرها است. مشکل سازگاری و ضمانت سازگاری تکثیرها از طریق اعمال سازگاری قوی بر طرف می‌شود. اما به دلیل ماهیت ذاتی تأخیر بالا در سیستم‌های ذخیره‌سازی موجود در سیستم‌های توزیعی استفاده از تکثیر همزمان و سازگاری قوی باعث افزایش ترافیک شبکه و هزینه‌ها و نهایتاً تأخیر در شبکه می‌شود. این تاخیرها هزینه‌های قابل توجه‌ای را به بار می‌آورند. به عنوان مثال هزینه تنها یک ساعت از کار افتادگی و تأخیر برای سیستم که مجوزهای فروش کارت اعتباری

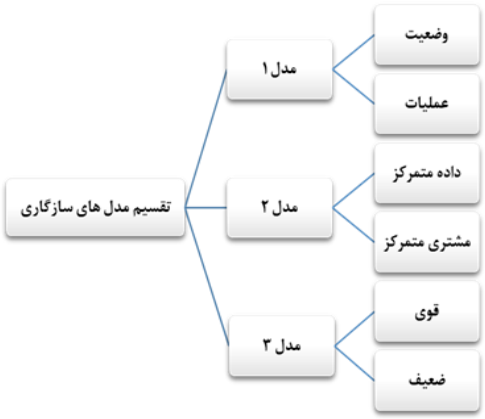
را صادر می‌کند، حدود ۳,۱ میلیون دلار است. لذا مدیریت سازگاری با حداقل تأخیر یک امر مهم و یک چالش در سیستم‌های توزیعی است [۲۵].

۳-۱- تئوری CAP [1] و چالش جدید سازگاری و کارایی

تئوری CAP به عنوان یک چالش در طراحی سیستم‌های ذخیره‌سازی مطرح شده است. در این تئوری اثبات شده است که تنها دو خصوصیت از سه خصوصیت (دسترسی‌پذیری، قابلیت حمل و) را می‌توان به طور عملی و همزمان فراهم کرد. در سیستم‌های توزیعی تضمین سازگاری و دسترسی‌پذیری با اهمیت‌تر است. در این میان به دلیل وجود تأخیر بالا در ماهیت سیستم‌های توزیعی ابر، محققان مصالحه‌ای جدید و مهمتری به نام سازگاری و کارایی را مطرح کردند که مدل‌های سازگاری متفاوتی توسط محققان برای غلبه بر این چالش‌ها مطرح شده است [۲۵].

۳-۲- تقسیم‌بندی موجود در سازگاری تکثیر

در منابع مختلف روش‌های سازگاری از نقطه نظرهای مختلفی تقسیم‌بندی شده‌اند. در شکل ۲ این تقسیم‌بندی‌ها به تصویر کشیده شده‌اند. در ادامه هر کدام را به تفصیل توضیح می‌دهیم. در تقسیم‌بندی اول مدل‌های سازگاری از نظر خصیصه‌های سازگاری به دو دسته کلی سازگاری وضعیت [2] و سازگاری عملیاتی [3] تقسیم می‌شود. در سازگاری وضعیت سازگاری بر روی وضعیت داده تمرکز دارد (مثلاً ناسازگاری دو تکثیر [7]، با انحراف عددی بیشتر از X، تعریف می‌شود). روش‌های موجود در دسته سازگاری وضعیت به دسته‌های نامتغیر [4]، محدوده‌ی خطا [5] تقسیم‌بندی شده است. روش‌هایی که در دسته سازگاری عملیاتی قرار می‌گیرند تمرکز بر روی عملیات‌ها و نتایجی که بر می‌گردانند است. این دسته خود به تعادل ترتیبی [6]، تعادل مراجع [7]، خواندن – نوشتن متمرکز [8]، تقسیم‌بندی می‌شود [۲۵- ۲۶]. از طرفی آقای اندرو تننبام مدل‌های سازگاری را قراردادی بین مراکز داده توزیع شده و پردازش‌ها دانسته که با رعایت آنها نتایج عملیات خواندن نوشتن درست و معتبر خواهد بود [۲۶].

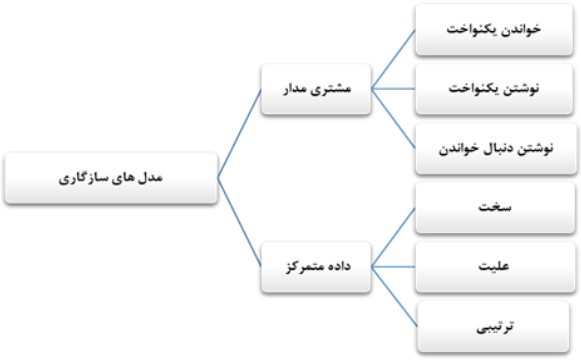


شکل ۲. تقسیم‌بندی‌های مرسوم مدل‌های سازگاری

مدل‌های سازگاری هر یک تعداد محدودی از نیازهای سازگاری را مطرح می‌کنند. بنابراین تا کنون مدل‌های بسیاری مطرح شده است. تقسیم‌بندی دوم توسط آقای تننبام ارائه شده است. در یک تقسیم‌بندی، تننبام مدل‌های سازگاری را در دو دسته داده-متمرکز [9] و مشتری- متمرکز [10] قرار می‌دهد. مدل داده- متمرکز بر روی سازگاری یک سیستم، از دیدگاه وسیع سیستم ذخیره‌سازی متمرکز می‌شود.

این تقسیم‌بندی در شکل ۳ به طور کامل‌تری همراه با مثال‌هایی از هر دسته به تصویر کشیده شده است. مثال‌های که در این مدل جا می‌گیرند عبارت‌اند از: سازگاری سخت، سازگاری علیت، سازگاری ترتیبی. مدل مشتری- متمرکز بر روی سازگاری از دیدگاه یک مشتری تمرکز دارد و تنها به داده‌های ذخیره شده به‌وسیله مشتری متمرکز می‌شود. مثال‌ها شامل: سازگاری خواندن یکنواخت [11]، سازگاری نوشتن یکنواخت [12]، سازگاری خواندن خود نوشته‌ها [13] و سازگاری نوشتن به دنبال خواندن [14].[۲۶]. در یک تقسیم‌بندی خیلی کلی‌تر مدل‌های سازگاری به دو دسته مدل‌های سازگاری قوی و مدل‌های سازگاری ضعیف تقسیم می‌شوند و مدل‌هایی مانند سازگاری شدید [15]، سازگاری ترتیبی [16]، سازگاری خطی‌سازی [17] در دسته قوی و مدل‌هایی مانند سازگاری احتمالی [18] و خواندن بعد از نوشتن [19] و خواندن بعد از نوشتن [20] و سازگاری علیت در دسته ضعیف قرار می‌گیرند [۲۶].

[1] Consistency, Availability, and Partition Tolerancey(CAP)
[2] State Consistency
[3] Operation Consistency
[4] Invariants
[5] Error Bounds
[6] Sequential Equivalence
[7] Reference Equivalence
[8] Read-Write Centric
[9] Data- Centric
[10] Client- Centric
[11] Monotonic Reads
[12] Monotonic write
[13] Read Your Writes
[14] Writes Follow Reads
[15] Strict Consistency
[16] Sequential Consistency
[17] Linearizability
[18] Eventual Consistency
[19] Read-After-Read
[20] Read After Write



شکل ۳. انواع مدل‌های سازگاری از دیدگاه تننبام

برخی از این مدل‌ها در جدول ۱ به اختصار توضیح داده شده‌اند. مدل‌های سازگاری مانند Fork، View، Adaptive [۳۶]، VFC3 [۳۷] و Red-Blue که نیازهای ضروری جدید سیستم‌های محبوب جدید مانند ابر، مه و اینترنت اشیاء صنعتی را برطرف می‌کنند، به عنوان مدل‌های جدید در نظر گرفته می‌شوند. برخی از نیازهای حیاتی جدید که این مدل‌ها بر روی آنها تمرکز دارند عبارتند از: امنیت، قابلیت اطمینان، تضمین SLA، همگرایی بیشتر در عملیات و مصرف انرژی. همچنین مدل‌های سازگاری مانند Time Casual، Fork و +Casual که از سایر مدل‌های پایه توسعه یافته‌اند، به عنوان مدل‌های توسعه‌یافته در نظر گرفته می‌شوند.

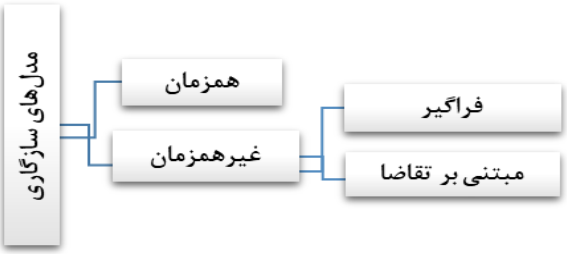
۳-۲-۱- مدل‌های سازگاری از منظر تکنیک به‌روزرسانی و شروع‌کننده انتشار

مدل‌های سازگاری از دیدگاه تکنیک که در به‌روزرسانی استفاده می‌کنند و از دیدگاه موجودیت شروع‌کننده به‌روزرسانی قابل بررسی و طبقه‌بندی است. در شکل ۴، ازنقطه‌نظر روش‌های به‌روزرسانی، سازگاری تکثیرها به دو روش همزمان و غیرهمزمان تقسیم می‌شوند [۲۳].

- روش‌های همزمان

در روش همزمان در هر بار عملیات نوشتن تمام تکثیرهای اصلی و ثانویه به‌روزرسانی می‌شوند. در این روش تا زمانی که تمام تکثیرها به‌روزرسانی نشود عملیات تکمیل‌شده در نظر گرفته نمی‌شود. درواقع این روش تراکنش‌ها به صورت واحد و اتمیک در نظر گرفته می‌شود. این روش با نام‌های به‌روز رسانی مشتاق و بدبینانه نیز شناخته می‌شوند. مزایا و معایب این دسته از روش‌ها عبارتند از:

مزایا: ۱. داده‌ها در هر زمان به‌روز هستند. ۲. هیچ برخوردی بین داده‌ها وجود ندارد.



شکل ۴. انواع مدل‌های سازگاری از منظر مدل‌های به‌روزرسانی

معایب: ۱. زمان پاسخ تراکنش‌ها را بالا می‌برد ۲. برای گره‌های متحرک مناسب نمی‌باشند. ۳. باید کل سیستم اصلی تا زمان به‌روزرسانی سایر تکثیرها منتظر بماند. ۴. درزمانی که تأخیر بالا و لینک‌های ارتباطی دارای احتمال خرابی هستند، این روش کارایی ندارد. ۵. درزمانی که تعداد تکثیرها و تعداد عملیات به‌روزرسانی بالا است سیستم به‌شدت با مشکل کارایی مواجهه می‌شود.

- روش‌های غیرهمزمان

در این روش‌ها تأکیدی بر دریافت تایدیه از نسخه‌های اصلی و ثانویه تکثیرها نداریم. همزمان‌سازی تکثیرها در بازه‌های زمانی انجام می‌شود. میزان این بازه بستگی به تأخیر لینک‌های ارتباطی و بار کاری سرورها دارد. در این مدل به‌روز رسانی باعث بلاک شدن دسترسی به سایر تکثیرها نمی‌شود. همزمان‌سازی تنبل و آرام شده[1] نام‌های دیگری برای به‌روزرسانی غیرهمزمان است. در سیستم‌های توزیع شده اغلب برنامه‌های توزیعی از این مدل به‌روزرسانی استفاده می‌کنند. این مدل به دو دسته فراگیر و مبتنی بر تقاضا تقسیم می‌شود. در دسته فراگیر، از یک فایل چندین نسخه از فایل همزمان می‌تواند وجود داشته باشد. وقتی یک فایل تغییر می‌کند به‌روزرسانی‌ها به شکل تنبل برای سایر تکثیرها به‌روزرسانی منتشر می‌شوند. در این وضعیت اغلب سطح ضعیفی از سازگاری را خواهیم داشت. در این روش هر تکثیر در هر زمانی می‌تواند به‌روزرسانی شود. لذا تعداد برخوردها زیاد خواهد بود. از این روش با نام ذخیره و ارسال[2] نیز یاد می‌شود. در دسته مبتنی بر تقاضا تا زمانی که کلاینتی در شبکه درخواست دسترسی به فایل را ندهد داده به روز نمی‌شود.

[1] Relaxed Synchronous

[2] Store And Forward

جدول ۱. شرح مختصری از برخی مدل‌های سازگاری رایج تقسیم‌بندی شده در پنج دسته داده متمرکز، مشتری متمرکز، ترکیبی، جدید و توسعه‌یافته

	داده متمرکز (data - centeric)
Strict[۲۷]	قوی ترین مدل سازگاری است. تکثیرها در سطح جهانی با زمان مطلق جهانی همگام می‌شوند.
Sequential [۲۶]	این مدل شبیه سازگاری strict است. با این حال، ضعیف‌تر است. فرآیندها عملیات نوشتن را مشاهده می‌کنند که دارای رابطه علیت با همان ترتیب هستند، اما عملیات خواندن اجرا شده توسط سایر فرآیندها قابل مشاهده نیست.
Causal [۱۷]	این مدل روابط علی بالقوه بین عملیات را تعیین می‌کند. این تضمین می‌کند که عملیات مرتبط با علت به همان ترتیب توسط همه فرآیندها بازدید می‌شود. بنابراین اگر عملیات خواندن نتیجه برخی از عملیات نوشتن باشد، تا زمانی که تمام عملیات نوشتن تمام نشده باشد، انجام نخواهد شد.
Weak [۲۶]	این مدل یک متغیر همگام‌سازی را به عنوان نشانه مشخص می‌کند. عملیات نوشتن و خواندن را می‌توان توسط فرآیندهایی که دارای این نشانه هستند اجرا کرد. در این مدل، داده‌های مشترک با سازگاری متوالی در متغیر همگام‌سازی قابل دسترسی است.
Realeses [۲۸]	از متغیر همگام‌سازی یا قفل استفاده می‌کند. این فرآیند، در مرحله اول، از یک قفل برای دسترسی به یک منبع مشترک (مثنی) درخواست می‌کند. سپس در مرحله دوم، قفل آزاد می‌شود و مقدار تغییر یافته ماکت به ماکت‌های دیگر ارسال می‌شود.
Realeas -lazy [۲۸]	این مدل یک بسط ضعیف قوام انتشار است. یک فرآیند می‌تواند نمایش نتایج عملیات نوشتن خود را تا عملیات همگام‌سازی بعدی به تاخیر بیندازد. از مهر زمانی استفاده می‌کند تا تشخیص دهد که نسخه جدیدترین به‌روزرسانی است یا قدیمی.
Entry [۲۹]	در این مدل قبل از دسترسی به هر داده، قفل آن دریافت می‌شود و سپس می‌توان هر عملیاتی را انجام داد.
	مشتری متمرکز (client centric)
Monotonic read [۲۶]	این مدل تضمین می‌کند که فرآیند هرگز ارزش قبلی مورد x را در زمانی که فرآیند مشاهده می‌کند مشاهده نکند. بانک اطلاعات ایمیل توزیع شده نمونه ای از این مدل است که در آن هر کاربر یک صندوق پستی در چندین نسخه دارد.
monotonic write [۲۶]	در این مدل، عملیات نوشتن بر روی یک داده توسط یک فرآیند زمانی قابل قبول است که عملیات نوشتن قبلی را به طور کامل روی آن انجام داده باشد.
Read Your Write [۲۶]	در این مدل، یک مشتری از هر مقدار نوشته شده جدید به ترتیب انجام توسط مشتری بازدید می‌کند. در واقع، پس از اتمام عملیات نوشتن قبلی، مشتری مقدار تازه نوشته شده را می‌خواند.
Write follow Read [۲۶]	این مدل به عنوان جلسه علیت نیز شناخته می‌شود. عملیات نوشتن روی یک آیتم داده زمانی توسط مشتری قابل قبول است که مشتری آخرین مقدار مورد-داده (item-data.) را خوانده باشد. توبیتر نمونه ای از استفاده از این مدل است.
	ترکیبی(Combination/ Hybrid)
Fork Sequential [۳۰]	در این مدل، مشتریان متعددی که می‌توانند یک تراکنش را به طور مستقیم یا غیرمستقیم مشاهده کنند، تمامی عملیات‌های قبلی را بر اساس تاریخچه رویداد خود مشاهده می‌کنند. این مدل اتمی بودن عملیات مشتری را تضمین نمی‌کند. این نقص در مدلی به نام Fork Linearize Consistency (FLC)) برطرف شده است.
Fork Linearize [۳۱]	
RedBlue Novel [۳۲]	این مدل عملیات را بر اساس نوع اجرا به عملیات قرمز و آبی تقسیم می‌کند. ترتیب اجرای عملیات قرمز باید در همه سایت‌ها یکسان باشد اما ترتیب رنگ آبی مهم نیست
	جدید
RedBlue [۳۲]	این مدل عملیات را بر اساس نوع اجرا به عملیات قرمز و آبی تقسیم می‌کند. ترتیب اجرای عملیات قرمز باید در همه سایت‌ها یکسان باشد اما ترتیب عملیات آبی مهم نیست
Timed [۳۳]	در این مدل، اگر عملیات نوشتن در زمان t انجام نشود، پس از زمان Δ، سایر گره‌ها قابل مشاهده هستند. این ترکیبی از ناسازگاری و ترتیب قدیمی را ارائه می‌دهد و به آن قوام دلتا می‌گویند.
Fork [۳۴]	برای سیستم‌های ذخیره‌سازی با کلاینت‌های غیر قابل اعتماد استفاده می‌شود و همزمانی و امنیت را افزایش می‌دهد. این سه ویژگی اصلی را اجرا می‌کند، از جمله ۱. کلاینت سالم لیستی از عملیات صحیح را به سرور ارسال می‌کند. ۲. یک کلاینت سالم تمام عملیات قبلی خود را مشاهده می‌کند. ۳. همه مراجعان سالم جنبه یکسانی از عملیات آن مشتری سالم دارند.
	توسعه یافته
Fork* [۳۵]	Fork* (FC) مشابه سازگاری fork است با این تفاوت که در خاصیت سوم، از ویژگی join-at -once استفاده می‌کند و دقت و بهبود بازرسی سیستم را بهبود می‌بخشد.

مزایا و معایب روش‌های غیرهمزمان عبارتند از:

مزایا:۱. برنامه‌ها نیاز به داشتن اطلاع از به‌روز رسانی‌ها و وجود تکثیرها در محل‌های دور ندارند، لذا کارایی کلی سیستم افزایش می‌یابد. ۲. در زمان به‌روزرسانی یک تکثیر سایر تکثیرهای همان قابل دسترس هستند. ۳. در زمان خراب شدن تکثیرها اصلی سایر تکثیرها قائل دسترس هستند. ۴. پایداری و دسترسی پذیری را تضمین می‌نماید. ۵. به ارمغان آوردن کاهش زمان دسترسی و افزایش کارایی به دلیل در دسترس بودن تکثیرها.

معایب: ۱. تعداد برخوردها در شرایطی که چند مستر داریم بالا می‌باشد ۲. افزایش برنامه‌های محاسباتی نیازمند به داده باعث وابستگی تنگاتنگ روش با محل تکثیر و وابستگی انتخاب تکثیر با کارایی به‌روزرسانی غیرهمزمان خواهد شد. ۳.مدلی ضعیف در تضمین سازگاری است. ۴. تعیین بازه زیاد یا کم منجر به افزایش سربار و یا کاهش درجه سازگاری خواهد شد.

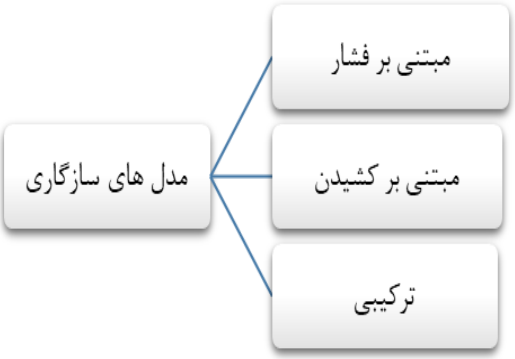
شکل ۵ تقسیم‌بندی مدل‌های سازگاری از نقطه نظر اینکه در انتشار به‌روز رسانی تکثیرها چه کسی شروع‌کننده است (سرور یا مشتری) روش‌های سازگاری موجود به سه دسته مبتنی بر کشیدن[1] و مبتنی بر فشاردادن[2]، ترکیبی[3] تقسیم‌بندی می‌شوند.

- روش‌های سازگاری مبتنی بر کشیدن (Pull-based)

سناریو این دسته مبتنی بر مشتری است به این معنی که مشتری با بیرون کشیدن داده‌ی به روز از سرور به سمت خود تغییرات را مقداردهی می‌کند (درخواست به سرور و انتقال از سرور). زمانی که تعداد به‌روزرسانی‌های کمتری در مقایسه با خواندن فایل توسط مشتری وجود دارد، از این روش می‌توان استفاده کرد. یکی از معایب این روش این است که هیچ به‌روزرسانی انجام نمی‌شود اگر سرور ازکار افتاده شود.

- روش‌های سازگاری مبتنی بر فشار (push- based)

در این روش سرور، داده‌های به روز را به مشتری‌ها منتشر می‌کند (بدون توجه به درخواست از سمت کلاینت). سرور در این مکانیسم‌ها نقش پررنگی در یکسان‌سازی تکثیرها دارد. زمانی که تعداد به‌روزرسانی و تعداد خواندن‌ها بالا می‌باشد این روش مفید و کاربردی است. زمانی که همه تکثیرها از سرور ارسال داده شوند زمان پاسخ تکثیرها ناچیز خواهد بود.



شکل ۵. انواع مدل‌های سازگاری از منظر شروع‌کننده بروز رسانی

- استراتژهای ترکیبی

در این استراتژی‌ها دو روش مبتنی بر فشار و مبتنی بر کشیدن را به صورت ترکیبی استفاده می‌کنند.

۴- روش‌های سازگاری تکثیر داده از منظر نوع سیستم‌های توزیعی

در این بخش به بررسی مدل‌های سازگاری در هردسته از خدمات توزیعی می‌پردازیم.

۴-۱- سازگاری در ابر محاسباتی

همانطور که محاسبات ابری به طور فزآینده‌ای محبوب می‌شود، همزمان خدمات ذخیره‌سازی ابر نیز به ایجاد امنیت، کارایی بالا و در دسترس بودن با هزینه کم برای کاربران خود بیشتر توجه دارند. انتظار می‌رود ذخیره‌سازی ابر به نیروی اصلی بازارهای ذخیره‌سازی آینده، تبدیل شود. همان‌طور که گفته‌شده تکنیک تکثیر داده یکی از تکنیک‌های محبوب در افزایش کارایی در سیستم‌های ابر است. تکثیر داده به‌عنوان یک تکنیک کلیدی در محاسبات ابری، با چالش سازگاری تکثیرها مواجهه می‌شود. در ابر مانند سایر سیستم‌های توزیعی، تحقق سازگاری تکثیرها به حل کردن همگام‌سازی بین دو تکثیر یا بیشتر می‌پردازد. در راستای برآورده کردن این هدف، سیستم با هزینه‌ها و چالش‌هایی مواجهه خواهد شد. محققان مدل‌های سازگاری تکثیر داده مختلفی را در تلاش برای برآورده کردن این اهداف در ابر ارائه کرده‌اند.

ازجمله سازگاری‌های پایه‌ای موجود در ابر، می‌توان به سازگاری قوی، ضعیف و سازگاری احتمالی اشاره کرد [۳۸]. در شکل مرسوم، سازگاری قوی (مشهور به سازگاری مشتاق[4])، با تغییر یک داده، تضمین می‌شود که همه تکثیرهای آن فوراً به‌روز رسانی شوند سپس به عملیات بعدی اجازه اجرا شدن داده شود. در نسخه دیگر از سازگاری قوی به نام سازگاری قوی- تنبل[5] در اولین درخواست خواندن، آخرین نسخه داده تغییر داده شده و به همه تکثیرهای موجود ارسال می‌شود. در سازگاری قوی هیچ تفاوتی بین تکثیرها وجود ندارد؛ بنابراین، در هر دسترسی تازه‌ترین داده به دست خواهد آمد؛ اما در این روش هزینه‌های نگهداری به طور قابل توجه‌ای افزایش می‌یابد، طوری که در دسترس بودن تکثیرها و عملکرد و کارایی سیستم را کاهش می‌دهد.

[1] Pull Based
[2] Push Based
[3] Hybrid
[4] Egger
[5] Lazy

در [۳۹]، کاری دیگر با موضوع سازگاری داده، مکانیسم تطبیقی و غیرهمزمان بر مبنای برنامه کاربردی برای سازگاری تکثیر در ابر پیشنهاد شد. مدیریت کپی‌ها در این مدل یک ساختار متمرکز دارد. دامنه اجرایی آن شامل یک گره اصلی، سه گره جانشین و گره‌های فرزند است. گره اصلی مهم‌ترین بخش از دامنه اجرایی است که اطلاعات همه گره‌ها را نگه می‌دارد و به طور همزمان، مسئول برای تأیید تمام درخواست‌های به‌روزرسانی است و سپس یک عملیات مربوطه را با توجه به استراتژی سازگاری تکثیر اجرا می‌کند. هدف آن‌ها حل کردن مشکلاتی مانند: تراکم شبکه و دسترسی‌پذیری پایین است. نتایج نشان می‌دهد که مکانیسم تطبیقی و ترکیبی پیشنهاد شده در این مقاله مقدار عملیات به‌روزرسانی را به طور چشمگیری کاهش می‌دهد. در شرایطی که به طور عمده نیاز برنامه‌های کاربردی از نقطه نظر سازگاری تضمین می‌شود. این روش غیرهمزمان و ترکیبی (مبتنی بر فشار و کشیدن) است. در این مقاله برای ارزیابی از شبیه‌ساز optorsim بهره برده شده است.

در [۴۰]، یک مدل تحلیلی سازگاری غیرهمزمان مبتنی بر صف‌بندی M/M/1 برای سیستم ذخیره‌سازی ابر خصوصی پیشنهاد شده است. در روش پیشنهادی برای رسیدگی کردن به تضادهای به‌روزرسانی، از روش احتمال حذف بر اساس درخواست‌های به روز شده استفاده شده است. علاوه بر این، یک روش دیگر نیز به منظور بهبود خوانایی پس از به‌روزرسانی و سازگاری ذخیره‌سازی در محیط ابر ارائه شده است.

در [۴۱]، یک روش جدید غیرهمزمان و ترکیبی در ابر به نام هارمونی پیشنهاد شده است که به صورت انطباقی سطح سازگاری را در زمان اجرا بر طبق نیازمندی‌های برنامه تعیین می‌کند. ایده اصلی پشت هارمونی ساخت یک مدل ارزیابی هوشمند از خواندن‌های قدیمی است. در نتیجه، سازگاری مورد نظر برنامه‌های کاربردی را در حال دستیابی به عملکرد خوب برآورده می‌کند. نتایج نشان داده که هارمونی می‌تواند به عملکردی خوب بدون تجاوز از تعداد قابل تحمل از خواندن‌های قدیمی دست یابد.

در [۴۲]، با استفاده از VFC[3] (Versatile Framework for Consistency)، یک مدل سازگاری غیرهمزمان جدید برای داده‌های تکثیر شده در سراسر مراکز داده ابر با چارچوب و پشتیبانی کتابخانه در پیشنهاد شده تا درجه زیادی از سازگاری را برای معانی مختلف داده به اجرا درآورد. این مقاله یک نمونه معماری از چارچوب VFC[3] را پیاده‌سازی و ارزیابی کرده و نتایج امیدوارکننده‌ای را آشکار کرده است. این روش در بهبود کیفیت خدمت، کاهش زمان تأخیر، پهنای باند و افزایش توان برای یک مجموعه‌ای از بارکاری کلیدی (و معمولی) مؤثر بوده است. این بهبود در حالی است که نیازها را در مورد کیفیت خدمات و اطلاعات ارائه شده به کاربران نهایی حفظ می‌کند.

در [۴۳]، نویسندگان استدلال می‌کنند که هزینه‌های مالی باید هنگام ارزیابی و انتخاب یک سطح سازگاری در ابر در نظر گرفته شوند. بر این اساس یک معیار جدید به نام بهره‌وری هزینه-سازگاری تعریف می‌کنند. بر اساس این معیار یک مدل سازگاری غیرهمزمان اقتصادی مؤثر ساده را به نام Bismar ارائه می‌دهند، که به صورت انطباقی سطح سازگاری را در زمان اجرا به منظور کاهش هزینه‌های مالی تنظیم می‌کند در حالی که به طور همزمان یک کسر کمی از خواندن‌های قدیمی را حفظ می‌کند. Bismar روی یک مدل احتمالی سازگاری تکیه می‌کند که خواندن‌های قدیمی و هزینه‌های مرتبط برنامه کاربردی را با توجه به نرخ خواندن/نوشتن فعلی و تأخیر شبکه تخمین می‌زند. Bismar در سیستم ذخیره‌سازی Cassandra روی Grid'5000 پیاده‌سازی شد. نتایج نشان داده روش مذکور هزینه‌ها را پایین نگه می‌دارد در شرایطی که تعداد خواندن‌های داده قدیم از تعداد قابل تحمل خواندن‌های قدیمی در برنامه‌های کاربردی تجاوز نمی‌کند..

در [۴۴]، یک سازگاری جدید در ابر به عنوان یک مدل خدمت به نامCaas ارائه شده است، که متشکل از یک ابر داده بزرگ و چندین ابر حسابرس کوچک است. در مدل Caas، یک ابر داده توسط یک ارائه‌دهنده خدمات ابر نگهداری می‌شود و یک گروه از کاربران که یک ابر حسابرس را تشکیل می‌دهند، می‌توانند تشخیص دهند آیا داده‌های ابر سطح وعده داده شده از سازگاری را ارائه می‌دهند یا خیر. در ادامه الگوریتم‌هایی برای تعیین کمیت شدت موارد نقض با دو معیار تشابه موارد نقض و کهنگی ارزش خواندن طراحی شده است.

در [۴۵]، الگوریتم‌های مؤثری را پیشنهاد کرده‌اند که به برخی سؤالات پاسخ می‌دهد. این سؤالات عبارتند از: چگونه یک کاربر می‌تواند بررسی کند که آیا یک سیستم ذخیره‌سازی وعده‌اش را از سازگاری برآورده می‌کند؟ تضمین تنها سازگاری احتمالی توسط یک سیستم چه مشکلاتی می‌تواند داشته باشد؟ الگوریتم‌ها با تجزیه و تحلیل ردی از تعاملات بین ماشین مشتری و یک انبار ارزش کلیدی، می‌توانند گزارش دهند که آیا رد، امن، منظم یا اتمیک است و اگر نیست چه تعداد موارد نقض وجود دارد. الگوریتم‌ها بر اساس رده‌هایی از کلیدهایی که بر اساس سازگاری شرطی ذخیره شده (Pahoehoe) ارزیابی شده‌اند. نتایج اجرا، تعدادی کم و در مواردی صفر مورد نقض را نشان داده است. به طور کلی این مقاله به چگونگی

ارزیابی و اندازه گیری سازگاری واقعی مشاهده شده به وسیله مشتری، هنگام استفاده از انبارهای ارزش کلیدی می‌پردازد.

در [۴۶]، سازگاری علیت برای سیستم‌های توزیعی بزرگ مقیاس ابر موردبررسی قرار گرفته است. اغلب کارهای پیشین سازگاری برای تکثیر کامل فایل مطرح شده است و تعداد کمی به سازگاری در حالت تکثیر جزیی پرداخته‌اند. در این مقاله دو الگوریتم ضعیف و تنبل برای حل چالش‌های موجود در این حیطه پیشنهاد شده است. این مقاله دو الگوریتم پیشنهادی خود را با نام‌های Full-Track و Opt-Track برای تکثیر داده به صورت جزئی ارائه داده‌اند. الگوریتم اول بهینه است. بهینگی از این جنبه است که در این الگوریتم، هر بار که علیت داده از بین می‌رود، داده‌ها در نزدیک‌ترین فاصله زمانی به‌روزرسانی می‌شوند. الگوریتم دوم نیز بهینه است به این معنی که اندازه فراداده حمل شده در پیام‌ها و ذخیره شده در سیاه‌های مربوطه به صورت محلی را کاهش می‌دهد. علاوه بر این، الگوریتم سومی مشتق شده از الگوریتم دوم ارائه می‌شود که سایز پیام، سربار، زمان پردازش و هزینه ذخیره‌سازی محلی در هر سایت را در حالت تکثیر کامل فایل را نیز کاهش می‌دهد. نتایج شبیه‌سازی حکایت از کاهش سربار در پرتوکل‌های سازگاری تکثیر جزئی (Opt-Track و Opt-Track) نسبت به پرتوکل‌های سازگاری تکثیر کامل (Opt-Track-CRP) در هزینه‌های انتقال و ذخیره‌سازی دارد. زمانی که سایرفایل‌ها بالا و میزان بارکاری نیز افزایش می‌یابد روش‌های بهینه (Opt-Track و Opt-Track) پیشنهادی در تکثیر جزئی موفقیت بیشتری از خود نشان داده‌اند.

در [۴۷]، یک الگوی تراکنش غیرهمزمان جدید برای سیستم توزیعی ابر ارائه شده است، که به طراحان این امکان را می‌دهد به تضمین سازگاری روی داده‌ها به جای سطح تراکنش بپردازند و همچنین روش پیشنهادی آنان این امکان را می‌دهد که تضمین سازگاری در زمان اجرا به صورت خودکار تغییر کند. آزمایش‌ها نشان داده که استراتژی تطبیقی ارائه شده در این مقاله به کاهش قابل توجه زمان پاسخ و هزینه‌ها از جمله هزینه جریمه ناسازگاری منجر می‌شود. این روش به طور قابل توجه‌ای هزینه‌های کلی را کاهش می‌دهد.

در [۴۸]، روشی به نام سازگاری علیت زمان‌بندی دقیق (STCC) را به عنوان یک مدل سازگاری ترکیبی در محیط ابر ارائه می‌کند. در این مدل سازگاری در دو سمت مشتری و سمت سرور در نظر گرفته شده است. در سمت مشتری، این مدل از روش‌های سازگاری خواندن یکنواخت، نوشتن یکنواخت، و نوشتن به دنبال خواندن پشتیبانی می‌کند. در سمت سرور، از سازگاری علّیت به موقع (TCC) نیز پشتیبانی می‌کند. علاوه بر این، از رویکرد مشتری متمرکز قوی‌تر است و نسبت به رویکردهای داده‌متمرکز انعطاف پذیرتر است. با وجود تحمل پارتیشن، روش پیشنهادی ثبات و در دسترس بودن داده‌ها را برآورده می‌کند. این مقاله به منظور ارزیابی بارهای کاری متفاوت، روی خوشه‌های کاساندرا اجرا شده است. نتایج تجربی مبتنی بر مقایسه بین روش پیشنهادی با ONE، ALL، QUORUM، و همچنین سازگاری‌های CC، در یک خوشه کاساندرا با ۲۴ گره، گواهی می‌دهد که رویکرد پیشنهادی به طور متوسط نرخ خواندن کهنه را تا ۲۴ درصد در حجم کار کاهش داده است.

توسعه سیستم‌های چند ابری با سطح قابل قبولی از تاخیرهای خدماتی که به طور قابل توجه‌ای بر کیفیت تجربه کاربران نهایی تأثیر می‌گذارد، یک مشکل تحقیقاتی باز است. در [۴۹]، یک مدل شبیه‌سازی سازگاری داده برای یک سیستم چند ابری با یک میان‌افزار توزیع‌شده جغرافیایی را در Matlab ارائه می‌شود. این روش به منظور تخمین زمان برای نوشتن به‌روزرسانی‌های داده‌ای که توسط کاربران نهایی در پایگاه‌های داده همه ارائه‌دهندگان خدمات ابری آغاز شده است پیشنهاد می‌شود. نتایج شبیه‌سازی نشان می‌دهد که مدل پیشنهادی می‌تواند در انتخاب یک پروتکل سازگاری داده‌ای که بر معماری یک سیستم چند ابری تأثیر می‌گذارد مفید باشد.

در [۵۰]، یک رویکرد سازگاری پویا، به نام سازگاری دینامیک مه ابری (CFDC[1]) پیشنهاد می‌شود. این روش مزایای رویکردهای خوش‌بینانه و بدبینانه را برای دستیابی و اطمینان از خدمات سازگاری محلی و جهانی در سیستم‌های ابر و مه ترکیب می‌کند. رویکرد CFDC پیشنهادی یک راه حل مناسب برای افزایش مقیاس محیط ابر و مه فراهم می‌کند تا عملکرد و کیفیت خدمات آن سیستم‌ها را به حداکثر برساند.

در مقاله [۵۱]، یک روش جایگذاری تکثیر پویا مبتنی بر ریسک اعتباری گره به نام RPCC در محیط ابر طراحی و اجرا شده است. هدف این روش استفاده از قرار دادن تکثیر برای متعادل کردن رابطه بین عملکرد سیستم و سازگاری تکثیر است. برای رسیدن به هدف، RPCC از روش تطبیقی غیرمتمرکز برای ایجاد و قرار دادن تکثیر به صورت پویا استفاده می‌کند. این روش فایل‌هایی با محبوبیت بالا و نرخ به‌روزرسانی پایین را برای ایجاد تکثیر در هنگام ایجاد کپی‌های جدید انتخاب می‌کند. در همین حال، نقشه ریسک اعتباری را با توجه به بار گره، درجه گره و تاخیر دسترسی ترسیم

[1] Cloud Fog Dynamic Consistency

می‌کند. RPCC گره‌های بهینه را برای قرار دادن تکثیرها بر اساس نقشه اعتباری انتخاب می‌کند. آزمایش‌های گسترده نشان می‌دهد که RPCC می‌تواند عملکردهای برتری را در افزایش احتمال به‌روزرسانی سازگاری به طور متوسط حدود ۵ درصد، کاهش تأخیر دسترسی حدود ۱۰ درصد و تعادل بار بالاتر نسبت به سایر روش‌های مشابه به دست آورد.

۴-۱-۱- محبوب‌ترین روش‌های سازگاری‌ها در ابر

در ابر روش‌های سازگاری متنوعی توسط محققان ارائه شده است. با این وجود روش‌های سازگاری مبتنی بر روش سازگاری نهایی یا احتمالی از محبوبیت بیشتری در ابر برخوردار هستند. در سازگاری احتمالی همه تکثیرها فوراً به‌روزرسانی نمی‌شود، بنابراین سیستم اختلاف تکثیرها را تحمل می‌کند؛ اما قول می‌دهد همه تکثیرها در یک زمان خاص سازگار باشند. در واقع سازگاری با تأخیر انداختن به‌روزرسانی‌ها، باعث افزایش سرعت دسترسی کاهش مصرف منابع شبکه می‌شود. در واقع نسبت به سازگاری‌های قوی هزینه‌ها به‌روزرسانی را کاهش می‌دهد.

به همین دلیل ارائه‌دهندگان ابر به منظور کاهش هزینه تمایل بیشتری به سازگاری نهایی دارند. سازگاری احتمالی اجازه می‌دهد سیستم برخی داده‌های قدیمی را برگرداند. اما نهایتاً تضمین می‌کند که داده‌ها در نهایت سازگار شوند. بسیاری از سیستم‌های امروزی مثل Google [۵۳]، Amazoon Dynamo [۲۹]، Cassandra [۵۲]، Big Table [۵۴]، Voledemort، Hbase [۵۵]، Riak [۵۶] مبتنی بر این راه حل‌های سازگاری نهایی هستند. البته این سیستم‌ها در شرایط خواندن و نوشتن‌های زیاد، عملکردشان به طور چشم‌گیری افت می‌کند که این میزان کاهش عملکرد در برخی از سیستم‌ها تا ۶۶,۶ درصد نیز گزارش شده است که بسیار نگران‌کننده است [۵۷]. سازگاری قوی و سازگاری احتمالی هر دو فقط برای شرایطی خاص مناسب هستند. در محیط محاسبات ابری، مشتریان ذخیره‌سازی در ابر، متنوع هستند. از طرفی برنامه‌های کاربردی به یک نوع ثابت و مشخص از سازگاری قوی یا سازگاری احتمالی، نیاز ندارند. نیاز سازگاری یک برنامه کاربردی در دوره‌های زمانی مختلف متغیر است. به عنوان مثال نرخ خواندن و نوشتن در یک فروشگاه اینترنتی، در بازه زمانی تعطیلات نسبت به زمان‌های دیگر بیشتر خواهد بود. لذا در این بازه به سطح سازگاری قوی‌تری نسبت به زمان‌های دیگر احساس می‌شود. استراتژی سازگاری تکثیر باید برای برنامه‌های کاربردی در شرایط مختلف مناسب باشد؛ و با تغییر الگوی دسترسی برنامه‌های کاربردی وفق‌پذیر باشد؛ بنابراین، سازگاری تطبیقی تکثیر بهترین انتخاب برای این محیط پویا خواهد بود. مدل‌های سازگاری‌های تطبیقی با توجه به شرایط مدل مناسب سازگاری تعیین می‌کنند.

۴-۲- سازگاری تکثیر داده در اینترنت اشیا و محاسبات مه

اینترنت اشیا شبکه‌ای از دستگاه‌های هوشمند متصل است. سنسورها، گوشی‌های هوشمند، دوربین‌ها و وسایل نقلیه نمونه‌هایی از این دستگاه‌های هوشمند هستند [۵۳] و [۵۸]. امروزه تعداد دستگاه‌های هوشمند و حجم داده‌های تولید شده در حال افزایش است. تخمین زده می‌شود که تعداد دستگاه‌های اینترنت اشیا (IoT) تا سال ۲۰۲۵ از ۷۵,۴۴ میلیون نفر فراتر رود [۵۹]. ابر طیف گسترده ای از نیازهای اینترنت اشیا را فراهم می‌کند. اما، بسیاری از برنامه‌های کاربردی در اینترنت اشیا نیاز به ذخیره و مدیریت داده‌های جریان، پردازش وظایف بلادرنگ، کاهش تأخیر، تضمین امنیت و مقیاس‌پذیری، و مدیریت تعداد زیادی از درخواست‌های مشتری حتی بدون استفاده از اینترنت دارند. رایانش ابری به دلیل متمرکز بودن، فاصله جغرافیایی بین منابع و درخواست‌کننده و حجم بالای ترافیک قادر به پاسخگویی به همه نیازهای برنامه‌های کاربردی در اینترنت اشیا نیست. به اینترنت هم نیاز دارد. امروزه، محاسبات مه به عنوان یک الگوی محاسباتی توزیع شده برای غلبه بر کاستی‌های ابر در حال ظهور است. مه خدمات را در لبه شبکه می‌آورد. بنابراین دسترسی به خدماتی با تاخیر کم را فراهم می‌کند. باید توجه داشته باشیم که مه جایگزینی برای ابر نیست. آن با چالش‌هایی مانند مدیریت منابع متنوع، امنیت، تأخیر، مقیاس‌پذیری، پویایی، پیچیدگی بالا، و ناهمگونی مواجهه است. از این رو، سازگاری مشابه در محاسبات مه یکی از مسائل مهم در عملکرد برنامه‌های کاربردی اینترنت اشیا است. از سوی دیگر، برنامه‌های کاربردی جدید در اینترنت اشیاء الزامات سازگاری جدیدی دارند. به عنوان مثال، اکثر راه حل‌های سازگاری فعلی در ابر، زمینه داده‌ها و مشتریان را در نظر نمی‌گیرند و فقط یک تنظیم سازگاری جهانی را ارائه می‌دهند. آنها تمام درخواست‌های مشتری را به عنوان یک جعبه سیاه در نظر می‌گیرند [۶۰]. در ابر، همه مشتریان دارای ضمانت‌های یکسان خواندن و نوشتن هستند. در مقابل در مه، این فرض ممکن است درست نباشد و زمینه مشتری سازگاری مورد نیاز را تعیین می‌کند. در برنامه مبتنی بر مه، امکان استفاده از زمینه، به ویژه مکان وجود دارد. باید بتواند سازگاری زمینه را در سطح عملیات (خواندن و نوشتن) به طور جداگانه فراهم کند. موقعیت مکانی یکی از حساس‌ترین کلاس‌های زمینه در مه

است [۶۱]. سازگاری منطقه‌ای نمونه‌ای از سازگاری مبتنی بر زمینه مکان است که اطلاعات مربوط به سطوح مورد نیاز سازگاری را جمع می‌کند. بر اساس این اطلاعات، سطح مناسبی از سازگاری ارائه می‌شود. برنامه‌های کاربردی ترافیک تمایل دارند از این ثبات استفاده کنند [۶۲].

در [۶۳]، یک طرح مدیریت سازگاری حافظه نهان مبتنی بر محبوبیت کارآمد را پیشنهاد شده است، که هدف آن تضمین تازگی داده‌های اینترنت اشیا بازگردانده شده توسط روترهای مسیریاب و جلوگیری از هزینه‌های سیگنالینگ سنگین معرفی شده در همان زمان است. شبیه‌سازی‌های گسترده‌ای تحت توپولوژی‌های بدون ترس و باینری درختی در دنیای واقعی انجام شد، و نتایج مربوطه کارایی طرح پیشنهادی را در حذف به‌موقع داده‌های اینترنت اشیاء منسوخ ذخیره‌شده توسط حافظه پنهان درون شبکه CCN ثابت کرده‌اند.

در [۸]، دو استراتژی برای مدیریت تکثیر و سازگار نگهداشتن داده‌های اینترنت اشیا و ثبات در زیرساخت‌های مه ارائه شده است. استراتژی‌ها برای هر داده، شماره تکثیر مناسب و مکان آن‌ها را انتخاب می‌کنند تا تأخیر دسترسی به داده‌ها و هزینه همگام‌سازی ماکت‌ها را کاهش دهند. این کار با رعایت سطح سازگاری لازم انجام می‌شود. همچنین، یک پلتفرم ارزیابی مبتنی بر شبیه‌ساز iFogSim را پیشنهاد می‌شود تا کاربران بتوانند استراتژی‌های خود را برای تکثیر داده‌های اینترنت اشیا و مدیریت سازگاری پیاده‌سازی و آزمایش کنند. آزمایش‌های نشان می‌دهد که هنگام استفاده از استراتژی‌های پیشنهادی، تأخیر خدمت را می‌توان تا ۳۰ درصد در مورد زیرساخت‌های Fog کوچک و ۱۳ درصد در مورد زیرساخت‌های مه در مقیاس بزرگ در مقایسه با iFogStor، کاهش داد. یک استراتژی پیشرفته و محبوب جاگذاری داده در زیرساخت مه است.

در[۶۴]، یک مدل مبتنی بر سازگاری علّیت برای ذخیره‌سازی توزیع شده بر اساس تکثیر جغرافیایی جزئی و ساختار همکاری به نام Cloud-Edge(PGCE) پیشنهاد شده است. این مدل مبتنی بر معماری شبکه توزیع شده همکاری Cloud-Edge است، و مجموعه داده‌های ابری توسط یک تابع هش به زیر مجموعه‌های متعدد تقسیم می‌شود تا این زیرمجموعه‌ها در گره‌های لبه‌ای که نزدیک به شبکه کاربر هستند برای تحقق بخشیدن به تکرار جغرافیایی جزئی، ذخیره شود. در همان زمان، مکانیسم تثبیت مهر زمانی و خدمت پردازش فراداده برای اجرای سازگاری داده‌ها بین گره‌ها با فرض اطمینان از علت، کاهش هزینه‌های سربار پردازش ابرداده و همگام‌سازی داده‌ها تنظیم شده‌اند. در این روش کلاینت مستقیماً با گره‌های لبه تعامل دارد که تأخیر پاسخ تعامل با ابر DC را کاهش می‌دهد. ارزیابی PGCE در مقایسه با مدل‌های موجود نشان می‌دهد که عملکرد بهتری در تأخیر پاسخ و توان عملیاتی دارد.

در [۶۵]، مدلی مشابه روش ارائه شده در [۶۴] ارائه شده است. این روش یک مدل سازگاری علّیت از همکاری لبه-ابر مبتنی بر پروتکل گروه‌بندی است.مراکز داده در این روش نیز با جدول هش پارتیشن‌بندی می‌شوند و تکثیر داده به صورت جزئی در گره‌های لبه انجام می‌شود. این روش در محیط همکاری edge-cloud در همان زمان، یک الگوریتم همگام‌سازی گروهی به نام Imp_Paxos اجرا می‌کند، در این روش به‌روزرسانی فقط نیاز به همگام‌سازی با گروه اصلی دارد که باعث کاهش تأخیر مشاهده به‌روزرسانی و کاهش سربار همگام‌سازی داده‌ها می‌شود.

در [۶۶]، یک روش مدیریت منابع و یک الگوریتم مدیریت تکثیر، که شامل تخصیص تکثیر و استراتژی حفظ سازگاری پیشنهاد شده است. در این روش‌ها با تکیه بر هزینه مالی به‌روزرسانی، سعی دارند هزینه‌های زمانی و هزینه از جمله هزینه‌های مالی سازگاری تکثیرها را برای مشتریان کاهش دهند. آزمایش‌ها نشان می‌دهد علاوه بر این الگوریتم مدیریت تکثیر به طور موثر زمان انتقال داده و سربار ذخیره‌سازی را کاهش دهد.

در [۵۰]، به منظور کاهش اتلاف منابع ذخیره‌سازی، یک استراتژی ایجاد تکثیر پویا بر اساس محبوبیت بلوک و بار گره (DRC-BN) پیشنهاد شده است. همچنین به منظور حل مشکل ناهماهنگی داده‌ها ناشی از همزمانی چند کاربر، یک استراتژی نگهداری سازگاری تکثیر بر اساس الگوریتم Fast Paxos (RC-FP) پیشنهاد شده‌است. استراتژی RC-FP عملکرد جامع گره‌ها را در نظر می‌گیرد و گره با عملکرد بهتر را با توجه به عملکرد جامع گره‌ها به عنوان رهبر انتخاب می‌کند و زمان نگهداری سازگاری را کاهش می‌دهد. نتایج تجربی نشان می‌دهد که استراتژی‌های پیشنهادی از نظر زمان نگهداری سازگاری و توان عملیاتی گره عملکرد خوبی دارند. در عین حال، استراتژی‌های پیشنهادی هزینه‌های کلی را نیز کاهش می‌دهند.

مقاله [۶۷]، به بررسی ایجاد تکثیر پویا، بازیابی سریع تکثیر و حفظ سازگاری تطبیقی در سیستم edg cloud می‌پردازد. قبل از حل مشکل ناسازگاری داده‌ها ناشی از به‌روز رسانی مکرر تکثیرها، استراتژی بازیابی سریع تکثیر را در نظر می‌گیرد. روش سازگاری پیشنهادی با نام ACP-IMP برای حل مشکل سازگاری شبیه‌سازی در محیط‌هایی با نرخ شکست بالای شبکه و گره پیشنهاد

شده است. در این روش گره‌های تکثیر با اطمینان بیشتر به عنوان رهبر گره‌های لبه انتخاب می‌شوند و زمان و تعداد انتقال پیام تایید مجدد کاهش می‌یابد. الگوریتم پیشنهادی به دلیل تاخیر پایین زمان دسترسی مناسب برای محیط‌های بلادرنگ است.

در [۷۵]، روشی برای مدیریت سازگاری نسخه در سیستم‌های با مقیاس بزرگ ارائه شده است، که ترکیبی از روش‌های خوش‌بینانه که کیفیت خدمت را تضمین می‌کنند و بدبینانه که بر بهبود کارایی تمرکز دارند می‌باشد، همچنین برای بهبود کیفیت الگوریتم از الگوریتم‌های جایگذاری مدل‌های بازار اقتصادی استفاده شده است. در این مدل گرید را به شکل سلسله مراتبی به دو سطح جداسازی می‌کنند. در سطح صفر گره‌های سایت قرار دارند که دارای منابع محاسباتی[1] و منابع ذخیره‌سازی هستند، در سطح یک مدیر نسخه‌ها قرار دارد که گره‌ای است که از بین گره‌های سایت انتخاب شده است. وظیفه مدیر نسخه‌ها حفظ سازگاری محلی درون سایت خودش و سازگاری سراسری همه‌ی نسخه‌ها با همکاری با سایر مدیر نسخه‌ها می‌باشد.

در [۳۴]، الگوریتم مدیریت سازگار موثری در گرید داده ارائه می‌شود که در آن داده‌های تکثیر را ثابت نگه می‌دارد. در این روش، گره‌ای که حاوی تکثیر است، وظیفه مدیریت سازگاری را بر عهده دارد. در هر گره، نرخ به‌روزرسانی تکثیر بر اساس میزان تغییر در فایل داده اصلی و نرخ درخواست دسترسی به نسخه مشابه موجود محاسبه می‌شود. مدل پیشنهادی تاخیرهای دسترسی را کاهش می‌دهد. نتایج تجربی نشان می‌دهد که مدل پیشنهادی در مقایسه با رویکردهای خوش بینانه و بدبینانه از نظر زمان اجرای کارو تعداد به‌روزرسانی‌ها عملکرد بهتری دارد.

در جدول ۲ مقایسه‌ای از برخی از مدل‌های سازگاری ارائه شده توسط محققان از منظر نوع سیستم توزیعی، دسته‌بندی مدل سازگاری ارائه شده، محیط پیاده‌سازی یا شبیه‌سازی، پارامترهای مهم دخالت داده شده در مدل سازگاری و پارامترهایی بهبود داده شده آورده شده است.

جدول ۲. مقایسه مدل‌های سازگاری پیشین.

پارامترهای بهبود داده شده	پارامترهای لحاظ شده در تصمیم‌گیری	محیط پیاده‌سازی یا شبیه‌ساز	دسته‌ی مدل سازگاری	نوع سیستم توزیعی	شماره مرجع
نرخ پرسش معتبر اما غلط[2]، نرخ دانلود معتبر اما غلط [3]	تعداد به‌روزرسانی‌ها وضعیت قطعی و وصلی گره‌ها	شبیه‌ساز جدید مبتنی بر رویداد (Event Based)	ترکیبی، تنبل، غیرهمزمان، توزیع شده	نظیر به نظیر	[۶۸]
زمان پاسخ، تعداد برخوردها	فرکانس دسترسی‌ها	شبیه‌ساز Optorsim	تنبل، نیمه‌متمرکز، غیرهمزمان، ترکیبی، (ترکیبی از سازگاری ضعیف و قوی)	گرید داده	[۷۰]
مدت زمان تکثیر	فرکانس دستیابی به فایل و ظرفیت رسانه ذخیره‌سازی	شبیه‌ساز Optorsim	غیرهمزمان، ترکیبی، (ترکیبی از سازگاری قوی و ضعیف)	گرید داده	[۷۲]
زمان پاسخ	درجه همسایگی گره‌های در ساخت درخت انتشار	شبیه‌ساز Optorsim	مبتنی بر فشار، نیمه متمرکز، همزمان، سازگاری قوی	گرید داده	[۵۴]
تعداد واگرایی‌ها و زمان پاسخ	بارکاری و کیفیت خدمت	شبیه‌ساز به زبان جاوا	غیرهمزمان، (ترکیبی از قوی و ضعیف)	گرید داده	[۷۳]
هزینه در هر تراکنش[4]،هزینه در زمان اجرا در هر تراکنش، هزینه سراسری در هر تراکنش	احتمال برخورد به‌روزرسانی‌ها در هر رکورد، تعداد به‌روزرسانی هزینه به‌روزرسانی	پیاده‌سازی بر روی سیستم ذخیره‌سازی Amazon's Simple Storage Service (S3)	غیرهمزمان، ترکیبی از سازگاری قوی و ضعیف	ابر محاسباتی	[۴۷]
درصد دسترسی به داده به‌روز، میزان حجم تراکنش‌ها،	فرکانس به‌روز رسانی، فرکانس دسترسی، نوع تکثیر(اصلی یا ثانویه)	شبیه‌ساز Optorsim	غیرهمزمان، ترکیبی،(ترکیبی از قوی و ضعیف)، تنبل	ابر محاسباتی	[۳۹]
احتمال خواندن‌های درست	متوسط نرخ خدمت، نرخ به‌روزرسانی، تعداد تکثیرها	مدل ارزیابی مبتنی بر صف‌بندی MMI	تنبل، ضعیف،غیرهمزمان	ابر محاسباتی	[۴۰]

[1] Computing Element
[2] Query False Valid Rate
[3] Downloading Query False Valid Rate
[4] Cost Per Transaction

کارایی حافظه نهان، کارایی در میزان تأخیر و گذردهی، توان عملیاتی، میزان مصرف پهنای‌باند	حداکثر تفاوت نسبی در محتویات دو داده شی داده، مدت زمانی که تکثیر می‌تواند داده قدیمی را تحمل کند، تعداد به‌روزرسانی‌های ممکن برای یکشی بدون به روز کردن تکثیرها	یک نمونه اولیه پیاده‌سازی شده به زبان جاوا	غیرهمزمان، (ترکیبی از قوی و ضعیف)، تنبل	ابر محاسباتی	[۴۲]
بهره‌وری هزینه سازگاری[1] ،هزینه مالی و نرخ خواندن‌های تازه، هزینه، نرخ خواندن داده‌های قدیمی	تأخیر شبکه ، نرخ خواندن و نوشتن	پیاده‌سازی بر روی سیستم ذخیره‌سازی Cassandra	غیرهمزمان، (ترکیبی از قوی و ضعیف)، تنبل	ابر محاسباتی	[۴۳]
سربار تبادل پیام، زمان انتقال، مدت زمان میانچی گری، پیچیدگی زمانی و فضایی	کلاک سیستم، آخرین زمان نوشتن	ذکر نشده	غیرهمزمان، ضعیف، ترکیبی، تنبل	ابر محاسباتی	[۵۷]
توان عملیاتی؛ تأخیر خواندن، تعداد خواندن‌های قدیمی،	نرخ تحمل خواندن داده‌های قدیمی توجه به داده‌های قدیمی و ساخت مدل مبنی بر این خواندن‌های قدیمی و مبتنی بر نیاز برنامه ارائه یک مدل ارزیابی هوشمند	در محیط Grid’5000 و Amazon EC2	غیرهمزمان، (ترکیبی از قوی و ضعیف)، تنبل	ابر محاسباتی	[۴۱]
نرخ پیام‌های به‌روز، زمان به‌روزرسانی، نرخ موفقیت به‌روزرسانی	نرخ خواندن و نوشتن، فروم‌های به‌روزرسانی	Peersim	غیرهمزمان، (ترکیبی از قوی و ضعیف)، تنبل	نظیر به نظیر	[۷۱]
میانگین زمان اجرای درخواست‌ها،	نرخ درخواست، زمان درخواست‌های قبلی، آخرین زمان درخواست داده	Optorsim	غیرهمزمان، (ترکیبی از قوی و ضعیف)، تنبل	گرید داده	[۳۴]
میانگین دسترسی-پذیری،زمان پاسخ، تعادل بار، زمان اتمام کار	تعداد پیام‌ها، هزینه ارتباط ورزولوشن‌ها، زمان خدمت	متلب	غیرهمزمان	لبه-ابر	[۶۷]
تاخیر عملیات خواندن، توان عملیاتی، بار کاری، تعتداد خواندن‌های داده‌های قدیمی	برچسب زمانی خواند و برچسب زمانی نوشتن	خوشه‌های کاساندرا	غیر همزمان، ضعیف	ابر	[۴۸]
هزینه‌های سربار پردازش ابرداده و همگام‌سازی داده‌ها، تاخیر زمان پاسخ	مهر زمانی، فاصله، علیت	شبیه ساز کلودسیم	غیر همزمان، ضعیف	ابر- لبه	[۶۴]
توان عملیاتی، تأخیر عملیات و تأخیر دید به‌روزرسانی	علیت، تثبیت مهر زمانی،	شبیه ساز کلودسیم	ترکیبی، ضعیف	ابر-لبه	[۶۵]
میانگین زمان بررسی ثبات و بازیابی. میانگین زمان حفظ سازگاری، تعداد دسترسیها، تعداد تکثیرها، زمان انتقال داده	منشا داده- زمان عملیات نوشتن، اطلاعات به‌روزرسانی	پیاده‌سازی در مراکز داده Hangzhou	غیر همزمان، ضعیف	ابر-لبه	[۶۶]
Load balancing and QoS	آخرین زمان به‌روزرسانی	شبیه‌سازی به زبان جاوا در Gloubos	قوی-ضعیف-همزمان غیر همزمان	گرید	[۷۴]
نسبت بار متوسط، احتمال آپدیت موفقیت آمیز،	متوسط تاخیر درخواست، درجه گره، همسایگی	OptorSim	غیر همزمان-قوی	ابر-لبه	[۶۷]
تعداد خواندن‌های ناسازگار، میانگین تعداد موارد کیفیت خدمت برآورده شده	مهر زمانی(آخرین زمان به‌روزرسانی)	شبیه ساز CloudSim	غیر همزمان، ضعیف	ابر- مه	[۵۰]

[1] Consistency-Cost Efficiency

۵- موضوعات باز و کارهای آتی

اکثر کارهای ارائه شده با موضوع مدل‌های سازگاری، اغلب در شبیه‌سازها مورد تست و ارزیابی قرارگرفته‌اند. ارزیابی روش‌های پیشنهادی در حیطه سازگاری در محیط‌های واقعی سیستم توزیعی یکی از کارهای آتی است که مورد توجه محققان در این حیطه است. اکثر کارهای ارائه شده به اندازه کافی سازگار و وفق‌پذیر با محیط نیستند. ارائه مدل‌های سازگارتر منعطف‌تر و وفق‌پذیرتر با محیط پویای سیستم‌های توزیعی یکی دیگر از مباحث نیمه کاره و مطلوب محققان این حیطه است. اغلب کارهای ارائه شده هزینه مالی را برای درخواست‌کنندگان سازگاری مد نظر قرار نداده‌اند. در کارهای پیشین اغلب مدل ارائه شده تک سروری بوده‌اند و تعداد کمی از کارها به مدل‌های چند سروی پرداخته‌اند. وجود چند سرور تکثیر باعث برخورد داده خواهد شد. پرداختن به این مسئله یکی دیگر موضوعات باز در این حیطه است. ترکیب الگوریتم‌های تکثیر پویا با مدل‌های سازگاری که هر دو با هم به بهترین نحوه عمل کنند یکی دیگر از موضوعات در کارهای آتی می‌تواند باشد. چراکه اغلب الگوریتم‌های تکثیر فرض کرده‌اند داده‌ها فقط خواندنی هستند و همچنین اغلب مدل‌های سازگاری پیشین مستقل از الگوریتم تکثیر عمل می‌کنند. برنامه‌های کاربردی نیازمند سطوح مختلفی از سازگاری هستند. لذا توجه به پارامترهای مختلف در برآورده کردن نیازی‌های سازگاری برنامه‌های کاربردی در تعیین سطوح سازگاری برای داده‌های برنامه کاربردی یکی دیگر شکاف‌ها در کارهای پیشین است. یکی از مسایلی که از دید محققان در کارهای پیشین پنهان مانده است; اهمیت توجه به محبوبیت یک فایل و تعیین سطوح سازگاری بر اساس تخمین محبوبیت یک فایل در زمان آتی است. در واقع فایل‌های محبوب به دلیل نیاز بیشتر در دسترسی به نسخه به روز آنها، نیازمند سازگاری قوی‌تری هستند. در واقع با پیشگویی دسترسی آتی به یک فایل می‌توان احتمال نیاز به سازگاری قوی برای آن فایل را پیشگویی کرد و با به روز کردن نسخه‌ی محبوب زمان پاسخ و کارایی سیستم را افزایش داد. در کارهای پیشین در تعیین سطوح سازگاری برای یک فایل میزان محبوبیت فایل‌ها در سطح یک فایل مطرح شده است. زمانی که یک تکثیر در یک سایت نسبت به سایر تکثیرهای آن فایل بیشتر مورد دسترس قرار می‌گیرد، دادن سطوح سازگاری قوی‌تر می‌تواند موجب افزایش کارایی شود. توجه به محبوبیت تکثیر فایل در یک سایت خاص به صورت جدا در عوض محبوبیت فایل در کل سیستم می‌تواند باعث کاهش دسترسی به داده قدیمی شود. لذا توجه بیشتر به محبوبیت تکثیرها به صورت جدا در مدل‌های پیشنهادی می‌تواند از جمله کارهای آتی در نظر گرفته شود. در تعداد کمی از کارها مدل سازگاری مبتنی بر یک مدل ریاضی بیان شده است. با توجه به اعتبار قطعی مدل ریاضی، ارائه روش‌های سازگاری مبتنی بر مدل‌های ریاضی می‌تواند به عنوان کارهای آتی در نظر گرفته شود.

امروزه به کاربران به ازای پرداخت هزینه، خدمت ارائه می‌شود. لذا ارایه مدل‌های مبتنی بر پرداخت هزینه را نیز می‌توان از چالش‌های پیشرو در نظر گرفته و در پژوهش‌های آتی مورد بررسی قرار داد. در سیستم‌های توزیعی مدرن مانند ابر ارائه خدمات مبتنی بر پرداخت هزینه مالی است. در واقع کاربران خدمات خود را از فراهم‌کنندگان اجاره می‌کنند. از این رو خدمت باید طوری ارائه شوند که هم برای اجاره‌کننده کم هزینه و هم برای فراهم‌کننده خدمت سودآور باشد. از این رو ارائه مدل‌های سازگاری که به این دو هزینه به طور همزمان توجه کنند از کارهای آتی در این حوزه است.

در تعداد کمی از کارهای قبلی، کیفیت خدمات (QoS) در نظر گرفته شده است. اکثر کارهای ارائه شده فقط زمان پاسخ را به عنوان معیار QoS در نظر می‌گیرند. ارائه روش‌های مدیریت سازگاری که به طور همزمان فهرستی جامع از پارامترهای کیفیت خدمات مانند توان عملیاتی، در دسترس بودن، مقیاس پذیری، امنیت، مصرف انرژی و قابلیت اطمینان را در نظر می‌گیرند در آینده می‌تواند بسیار جالب توجه باشد.

۶- نتیجه‌گیری

محبوبیت سیستم‌های توزیعی روز به روز در حال افزایش است. از طرفی حجم داده‌های تولیدی توسط برنامه‌های کاربردی در سیستم‌های توزیعی و میزان نیاز به دسترسی مؤثر و کارا به داده‌ها نیز در حال افزایش است. تکثیر داده تکنیکی کلیدی برای رسیدن به کارایی و دسترسی‌پذیری است. داده‌ها در سیستم‌های توزیعی توسط کاربران و برنامه‌ها دائما در حال تغییراند. از این رو حفظ و برقراری سازگاری تکثیرهای داده در این سیستم‌ها یک نیاز حیاتی است. از گذشته دور تا به اکنون، تلاش‌های قابل توجه‌ای برای توسعه استراتژی‌های سازگاری مختلف اختصاص یافته است. نتیجه این تلاش‌ها ظهور استراتژی‌های مختلف بوده است.

در این مقاله یک بررسی کلی در مورد استراتژی‌های سازگاری تکثیر داده در سیستم‌های توزیعی محبوب از جمله گرید، ابر، مه، لبه، نظیر به نظیر انجام شده است. تقسیم‌بندی‌های الگوریتم تکثیر از دیدگاه نوع تکثیر، بخش‌های مدیریت تکثیر، محیط تکثیر، هدف الگوریتم تکثیر و روش ارزیابی الگوریتم تکثیر بیان شد. با توجه به تمرکز بر روی سازگاری تکثیر، مفهوم سازگاری تکثیر بررسی شد و

انواع تقسیم‌بندی‌های موجود از دیدگاه‌هایِ مختلف از جمله خصوصیت، دیدگاه آقای تننبام، میزان همسان بودن نسخه بیان شد. از طرفی در یک طبقه‌بندی دقیق‌تر، ما مدل‌های سازگاری‌های پایه را به پنج دسته داده-متمرکز، مشتری-متمرکز، ترکیبی، جدید و بسط یافته تقسیم کردیم. تقسیم‌بندی مدل‌های سازگاری از نقطه نظر تکنیک به‌روزرسانی ذکر شد. در این بررسی مدل‌های سازگاری به دو دسته همزمان و غیرهمزمان تقسیم شدند. همچنین از نقطه نظر اینکه در انتشار به‌روز رسانی تکثیرها چه کسی شروع‌کننده بوده است (سرور یا مشتری)، روش‌های سازگاری موجود به سه دسته مبتنی بر کشیدن و مبتنی بر فشاردادن، ترکیبی تقسیم‌بندی شدند. علاوه بر این در هر نوع سیستم توزیعی مدل‌های سازگاری ارائه شده توسط محققان مطرح شدند و در جدولی برخی از مدل‌های سازگاری ارائه شده از منظر نوع سیستم توزیعی، دسته‌بندی مدل سازگاری ارائه شده، محیط پیاده‌سازی و شبیه‌سازی، پارامترهای مهم دخالت داده شده در مدل سازگاری و پارامترهایی بهبود داده شده مقایسه شدند.

بررسی‌ها نشان داد مدل‌های پیشنهادی محققان در تمام سیستم‌های توزیعی اغلب به یک یا چند مورد از موارد زیر می‌پرداختند: قابلیت اطمینان، مقیاس‌پذیری، تحمل خطا، متعادل‌سازی بار و غیره. این استراتژی‌ها سعی می‌کردند پارامترهای خاصی مانند تازگی (قدیمی نبودن)، گستره ساخت، هزینه ارتباط، هزینه به‌روزرسانی، زمان پاسخ‌دهی، مصرف پهنای‌باند، تأخیر دسترسی، متعادل‌سازی بار، هزینه تعمیر و نگهداری، زمان اجرای کار، تحمل خطا، و قرار دادن تکثیر استراتژیک را بهبود بخشند. هدف اکثریت استراتژی‌های همگام‌سازی یا سازگاری تکثیر داده دستیابی به صحت و کیفیت خدمات تکثیرهای داده است. همچنین بررسی‌ها نشان میدهد به دلیل هزینه‌های بالا سازگاری‌های قوی، تمایل به سمت استفاده از سازگاری‌های ضغیف در سیستم‌های توزیعی مانند ابر بسیار بالا است. البته میزان کاهش مطلوب سطح سازگاری تا حد بسیار زیادی بستگی به الگوهای دسترسی برنامه کاربردی و نرخ به‌روزرسانی داده‌های تکثیر شده و همچنین هدف به کار گیری آن داده‌ها دارد. لذا خدمت‌گیرندگان نیازمند سطوح سازگاری متفاوتی هستند. در این مقاله در بخشی جدا به کارهای آتی در این حوزه پرداخته شد. مدیریت سازگاری تکثیر مبتنی بر کیفیت خدمت، مدیریت سازگاری تکثیر مبتنی بر هزینه‌های مالی، مدیریت پویای سازگاری تکثیر مبتنی بر نیاز پویای برنامه‌های کاربردی، برخی از این کارهای آتی مطرح شده است.

مراجع

[1] R. Moore, C. Baru, A. Rajasekar, R. Marciano, and M. Wan, “Data Intensive Computing, In``The Grid: Blueprint for a New Computing Infrastructure’’, eds. I. Foster and C. Kesselman.” Morgan Kaufmann, San Francisco, 1999.

[2] M. Beigrezaei, A. Toroghi Haghighat, and S. Leili Mirtaheri, “Minimizing data access latency in data grids by neighborhood-based data replication and job scheduling,” *Int. J. Commun. Syst.*, p

[3] A. M. Rahmani, Z. Fadaie, and A. T. Chronopoulos, “Data placement using Dewey Encoding in a hierarchical data grid,” *J. Netw. Comput. Appl.*, vol. 49, pp. 88–98, 2015.

[4] U. Tos, R. Mokadem, A. Hameurlain, and T. Ayav, “Achieving query performance in the cloud via a cost-effective data replication strategy,” *Soft Comput.*, vol. 25, no. 7, pp. 5437–5454, 2021.

[5] X. Dong, J. Li, Z. Wu, D. Zhang, and J. Xu, “On dynamic replication strategies in data service grids,” in *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*, 2008, pp. 155–161.

[6] M.-C. Lee, F.-Y. Leu, and Y. Chen, “PFRF: An adaptive data replication algorithm based on star-topology data grids,” *Futur. Gener. Comput. Syst.*, vol. 28, no. 7, pp. 1045–1057, 2012.

[7] C. Li, M. Song, M. Zhang, and Y. Luo, “Effective replica management for improving reliability and availability in edge-cloud computing environment,” *J. Parallel Distrib. Comput.*, 2020.

[8] M. I. Naas, L. Lemarchand, P. Raipin, and J. Boukhobza, “IoT data replication and consistency management in fog computing,” *J. Grid Comput.*, vol. 19, no. 3, pp. 1–25, 2021.

[9] S. Sun, W. Yao, B. Qiao, M. Zong, X. He, and X. Li, “RRSD: A file replication method for ensuring data reliability and reducing storage consumption in a dynamic Cloud-P2P environment,” *Futur. Gener. Comput. Syst.*, vol. 100, pp. 844–858, 2019.

[10] M. Beigrezaei, A. T. Haghighat, and H. R. Kanan, “A new fuzzy based dynamic data replication algorithm in data grids,” in *2013 13th Iranian Conference on Fuzzy Systems (IFSC)*, 2013, pp. 1–5, doi: 10.1109/IFSC.2013.6675676.

[11] K. Rajaretnam, M. Rajkumar, and R. Venkatesan, “Rplb: A replica placement algorithm in data grid with load balancing,” *Int. Arab J. Inf. Technol.*, vol. 13, no. 6, 2016.

[12] H. K. H. So and R. Brodersen, “A unified hardware/software runtime environment for FPGA-based reconfigurable computers using BORPH,” *Trans. Embed. Comput. Syst.*, vol. 7, no. 2, pp. 1–28, 2008, doi: 10.1145/1331331.1331338.

[13] A. M. Rahmani, L. Azari, and H. A. Daniel, “A file group data replication algorithm for data grids,” *J. Grid Comput.*, vol. 15, no. 3, pp. 379–393, 2017.

[14] G. Hager and G. Wellein, *Introduction to high performance computing for scientists and engineers*. CRC Press, 2010.

[15] M. Beigrezaei, A. T. Haghighat, M. R. Meybodi, and M. Runiassy, “PPRA: A new pre-fetching and prediction based replication algorithm in data grid,” in *2016 6th International Conference on Computer and Knowledge Engineering (ICCKE)*, 2016, pp. 257–262.

[16] B. L. Chamberlain, D. Callahan, and H. P. Zima, “Parallel programmability and the chapel language,” *Int. J. High Perform. Comput. Appl.*, vol. 21, no. 3, pp. 291–312, 2007.

[17] L. Azari, A. M. Rahmani, H. A. Daniel, and N. N. Qader, “A data replication algorithm for groups of files in data grids,” *J. Parallel Distrib. Comput.*, vol. 113, pp. 115–126, 2018.

[18] Y. Saito, “Consistency management in optimistic replication algorithms,” *INTERNET, AOnlineU*, vol. 15, pp. 1–18, 2001.

[19] R. A. Campêlo, M. A. Casanova, D. O. Guedes, and A. H. F. Laender, "A brief survey on replica consistency in cloud environments," *J. Internet Serv. Appl.*, vol. 11, no. 1, pp. 1–13, 2020.

[20] J. Curtis, "Consistency of data replication protocols in database systems: a review," 2014.

[21] P. Vashisht, A. Sharma, and R. Kumar, "Strategies for replica consistency in data grid–a comprehensive survey," *Concurr. Comput. Pract. Exp.*, vol. 29, no. 4, p. e3907, 2017.

[22] K. S. Maabreh, "An Enhanced University Registration Model Using Distributed Database Schema.," *KSII Trans. Internet Inf. Syst.*, vol. 13, no. 7, 2019.

[23] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the grid: Enabling scalable virtual organizations," *Int. J. High Perform. Comput. Appl.*, vol. 15, no. 3, pp. 200–222, 2001.

[24] H. Yu and A. Vahdat, "Design and evaluation of a conit-based continuous consistency model for replicated services," *ACM Trans. Comput. Syst.*, vol. 20, no. 3, pp. 239–282, 2002.

[25] T. Kraska, M. Hentschel, G. Alonso, and D. Kossmann, "Consistency rationing in the cloud: pay only when it matters," *Proc. VLDB Endow.*, vol. 2, no. 1, pp. 253–264, 2009.

[26] A. S. Tanenbaum and M. Van Steen, *Distributed systems: principles and paradigms*. Prentice-Hall, 2007.

[27] J. Du, S. Elnikety, and W. Zwaenepoel, "Clock-SI: Snapshot isolation for partitioned data stores using loosely synchronized clocks," in *2013 IEEE 32nd International Symposium on Reliable Distributed Systems*, 2013, pp. 173–184.

[28] P. Keleher, A. L. Cox, and W. Zwaenepoel, "Lazy release consistency for software distributed shared memory," *ACM SIGARCH Comput. Archit. News*, vol. 20, no. 2, pp. 13–21, 1992.

[29] L. Iftode, J. P. Singh, and K. Li, "Scope consistency: A bridge between release consistency and entry consistency," in *Proceedings of the eighth annual ACM symposium on Parallel algorithms and architectures*, 1996, pp. 277–287.

[30] C. Cachin, I. Keidar, and A. Shraer, "Fork sequential consistency is blocking," *Inf. Process. Lett.*, vol. 109, no. 7, pp. 360–364, 2009.

[31] C. Cachin, A. Shelat, and A. Shraer, "Efficient fork-linearizable access to untrusted shared memory," in *Proceedings of the twenty-sixth annual ACM symposium on Principles of distributed computing*, 2007, pp. 129–138.

[32] C. Li, D. Porto, A. Clement, J. Gehrke, N. Preguiça, and R. Rodrigues, "Making geo-replicated systems fast as possible, consistent when necessary," in *Presented as part of the 10th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 12)*, 2012, pp. 265–278.

[33] [F. J. Torres-Rojas and E. Meneses, "Convergence through a weak consistency model: Timed causal consistency," *CLEI Electron. J*, vol. 8, no. 2, pp. 1–2, 2005.

[34] J. Li, M. N. Krohn, D. Mazieres, and D. E. Shasha, "Secure Untrusted Data Repository (SUNDR).," in *Osdi*, 2004, vol. 4, p. 9.

[35] A. J. Feldman, W. P. Zeller, M. J. Freedman, and E. W. Felten, "SPORC: Group collaboration using untrusted cloud resources," 2010.

[36] Y. Saito and M. Shapiro, "Optimistic replication," *ACM Comput. Surv.*, vol. 37, no. 1, pp. 42–81, 2005.

[37] M. Senftleben and K. Schneider, "Operational characterization of weak memory consistency models," in *International Conference on Architecture of Computing Systems*, 2018, pp. 195–208.

[38] [D. Terry, V. Prabhakaran, R. Kotla, M. Balakrishnan, and M. K. Aguilera, "Transactions with Consistency Choices on Geo-Replicated Cloud Storage."

[39] X. Wang, S. Yang, S. Wang, X. Niu, and J. Xu, "An application-based adaptive replica consistency for cloud storage," in *2010 Ninth International Conference on Grid and Cloud Computing*, 2010, pp. 13–17.

[40] Y. N. Aye, "Data Consistency on Private Cloud Storage System," *Int. J. Emerg. Trends Technol. Comput. Sci. Vol.*, vol. 1, 2012.

[41] H.-E. Chihoub, S. Ibrahim, G. Antoniu, and M. S. Perez, "Harmony: Towards automated self-adaptive consistency in cloud storage," in *2012 IEEE International Conference on Cluster Computing*, 2012, pp. 293–301.

[42] S. Esteves, J. Silva, and L. Veiga, "Quality-of-service for consistency of data geo-replication in cloud computing," in *European Conference on Parallel Processing*, 2012, pp. 285–297.

[43] H.-E. Chihoub, S. Ibrahim, G. Antoniu, and M. S. Perez, "Consistency in the cloud: When money does matter!," in *2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*, 2013, pp. 352–359.

[44] Q. Liu, G. Wang, and J. Wu, "Consistency as a service: Auditing cloud consistency," *IEEE Trans. Netw. Serv. Manag.*, vol. 11, no. 1, pp. 25–35, 2014.

[45] X. Ren, W. Ru-chuan, and K. Qiang, "Efficient model for replica consistency maintenance in data grids," in *International Symposium on Computer Science and its Applications*, 2008, pp. 159–162.

[46] A. Stiemer, I. Fetai, and H. Schuldt, "Analyzing the performance of data replication and data partitioning in the cloud: the BEOWULF approach," in *2016 IEEE international conference on big data (Big Data)*, 2016, pp. 2837–2846.

[47] T. Kraska, M. Hentschel, G. Alonso, and D. Kossmann, "Consistency rationing in the cloud: Pay only when it matters," *Proc. VLDB Endow.*, vol. 2, no. 1, pp. 253–264, 2009, doi: 10.14778/1687627.1687657.

[48] H. N. S. Aldin, H. Deldari, M. H. Moattar, and M. R. Ghods, "Strict timed causal consistency as a hybrid consistency model in the cloud environment," *Futur. Gener. Comput. Syst.*, vol. 105, pp. 259–274, 2020.

[49] O. Kozina and M. Kozin, "Simulation Model of Data Consistency Protocol for Multicloud Systems," in *2022 IEEE 3rd KhPI Week on Advanced Technology (KhPIWeek)*, 2022, pp. 1–4.

[50] [N. Mostafa, "A dynamic approach for consistency service in cloud and fog environment," in *2020 fifth international conference on fog and mobile edge computing (FMEC)*, 2020, pp. 28–33.

[51] S. Sun, X. Wang, and F. Zuo, "RPCC: A Replica Placement Method to Alleviate the Replica Consistency under Dynamic Cloud," in *2020 International Conferences on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics)*, 2020, pp. 729–734.

[52] A. Lakshman and P. Malik, "Cassandra: a decentralized structured storage system," *ACM SIGOPS Oper. Syst. Rev.*, vol. 44, no. 2, pp. 35–40, 2010.

[53] [OpenACC Working Group, "The OpenACC Application Programming Interface 3.0," *Openacc.Org Doc.*, pp. 1–118, 2019.

[54] C.-T. Yang, W.-C. Tsai, T.-T. Chen, and C.-H. Hsu, "A one-way file replica consistency model in data Grids," in *The 2nd IEEE Asia-Pacific Service Computing Conference (APSCC 2007)*, 2007, pp. 364–373.

[55] M. N. Vora, "Hadoop-HBase for large-scale data," in *Proceedings of 2011 International Conference on Computer Science and Network Technology*, 2011, vol. 1, pp. 601–605.

[56] [R. Klophaus, "Riak core: Building distributed applications without shared state," in *ACM SIGPLAN Commercial Users of Functional Programming*, 2010, p. 1.

experience in edge cloud computing system," *Inf. Sci. (Ny).*, vol. 516, pp. 33–55, 2020.

[67] J. Guo, C. Li, and Y. Luo, "Fast replica recovery and adaptive consistency preservation for edge cloud system," *Soft Comput.*, vol. 24, pp. 14943–14964, 2020.

[68] J. Lan, X. Liu, P. Shenoy, and K. Ramamritham, "Consistency maintenance in peer-to-peer file sharing networks," in *Proceedings the Third IEEE Workshop on Internet Applications. WIAPP 2003*, 2003, pp. 90–94.

[69] H. Shen, "IRM: Integrated file replication and consistency maintenance in P2P systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 1, pp. 100–113, 2009.

[70] R.-S. Chang and J.-S. Chang, "Adaptable replica consistency service for data grids," in *Third International Conference on Information Technology: New Generations (ITNG'06)*, 2006, pp. 646–651.

[71] [X. Meng and C. Zhang, "An ant colony model based replica consistency maintenance strategy in unstructured P2P networks," *Comput. Networks*, vol. 62, pp. 1–11, 2014.

[72] [S. C. Choi and H. Y. Youn, "Dynamic hybrid replication effectively combining tree and grid topology," *J. Supercomput.*, vol. 59, pp. 1289–1311, 2012.

[73] [E. Anderson, X. Li, M. A. Shah, J. Tucek, and J. J. Wylie, "What Consistency Does Your {Key-Value} Store Actually Provide?," 2010.

[74] G. Belalem and B. Yagoubi, *Collaborative Negotiation to Resolve Conflicts among Replicas in Data Grids*. INTECH Open Access Publisher, 2010.

[75] G. Belalem, C. Haddad, and Y. Slimani, "An effective approach for consistency management of replicas in Data Grid," in *2008 IEEE International Symposium on Parallel and Distributed Processing with Applications*, 2008, pp. 11–18.

[57] L. Xiong, L. Yang, Y. Tao, J. Xu, and L. Zhao, "Replication strategy for spatiotemporal data based on distributed caching system," *Sensors*, vol. 18, no. 1, p. 222, 2018.

[58] M. I. Naas, J. Boukhobza, P. R. Parvedy, and L. Lemarchand, "An extension to ifogsim to enable the design of data placement strategies," in *2018 IEEE 2nd International Conference on Fog and Edge Computing (ICFEC)*, 2018, pp. 1–8.

[59] R. Oma, S. Nakamura, D. Duolikun, T. Enokido, and M. Takizawa, "Fault-tolerant fog computing models in the IoT," in *International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, 2018, pp. 14–25.

[60] A. V. Dastjerdi, H. Gupta, R. N. Calheiros, S. K. Ghosh, and R. Buyya, "Fog computing: Principles, architectures, and applications," in *Internet of things*, Elsevier, 2016, pp. 61–75.

[61] M. Verma, N. Bhardwaj, and A. K. Yadav, "Real time efficient scheduling algorithm for load balancing in fog computing environment," *Int. J. Inf. Technol. Comput. Sci*, vol. 8, no. 4, pp. 1–10, 2016.

[62] H. F. Atlam, R. J. Walters, and G. B. Wills, "Fog computing and the internet of things: a review," *big data Cogn. Comput.*, vol. 2, no. 2, p. 10, 2018.

[63] B. Feng, A. Tian, S. Yu, J. Li, H. Zhou, and H. Zhang, "Efficient Cache Consistency Management for Transient IoT Data in Content-Centric Networking," *IEEE Internet Things J.*, 2022.

[64] [T. Junfeng, B. Wenqing, and J. Haoyi, "PGCE: A distributed storage causal consistency model based on partial geo-replication and cloud-edge collaboration architecture," *Comput. Networks*, vol. 212, p. 109065, 2022.

[65] J. Tian, H. Jia, and W. Bai, "CCECGP: causal consistency model of edge–cloud collaborative based on grouping protocol," *J. Supercomput.*, pp. 1–24, 2022.

[66] C. Li, J. Bai, Y. Chen, and Y. Luo, "Resource and replica management strategy for optimizing financial cost and user